ASCCBOT: AN OPEN MOBILE ROBOT PLATFORM

By

GANG LI

Master of Science in Electrical Engineering
Oklahoma State University
Stillwater, Oklahoma, USA
2011

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2011

ASCCBOT: AN OPEN MOBILE ROBOT PLATFORM

Thesis Approved:

Dr. Weihua Sheng

Thesis Advisor

Dr. Qi Cheng

Dr. Martin Hagan

Dr. Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this thesis, an open mobile robot platform is developed for mobile robotics research. In this chapter, the motivation and objectives of this thesis are presented as an overall introduction.

1.1 Motivation

The challenges of making robots really possess human-like intelligence, require decades of research in such fields as signal processing, statistics, machine learning and computer vision to reach the level where the developed algorithms can be applied in real-world, practical applications [3]. Although we are still far from building real intelligent robots due the challenges mentioned, we are closer than ever to fulfill the dream of mobile robots, which is described by Karel Capek: "Cheap, mobile intelligent machines present in every home" [4]. The idea of mobile robots has been further confirmed by the the development programs sponsored by industry as well as government agencies. As a matter of fact, robots have come into our lives in recent years. Some of these robots can be seen in Figure 1.1.

Mobile robotics apart from other robotics research areas such as conventional manipulator robotics, emphasizes on problems related to the understanding of large-scale space [5]. To understand a large-scale environment and behave intelligently inside not only implies mobility, but also requires the capability of identifying features, detecting patterns, learning from experience, localization, building maps, navigation, and most importantly, simultaneously exhibiting the above capabilities in concert [6].

Figure 1.1: Existing Mobile Robots

Robot simulation softwares, such as Microsoft Robotics Developer Studio [7], Webots [8] and Player/Stage/Gazebo [9], have been widely used in mobile robot research. But the robustness, efficiency and intelligence of the robot algorithms can only be thoroughly tested using the realtime sensory data from real-life environments. Therefore, there is a need to employ physical robot platforms for mobile robot research.

Multi-robot research is of great interest in recent years [10]. With features such as wireless communication, distributed computing, and mobility, mobile robot networks have great potential in many applications including security and military surveillance. In order to develop a networked mobile robot platform, we propose that each mobile robot node should meet the following criteria:

- Autonomy: Each node should have a reasonable amount of computational resource including CPU power and memory for its autonomous navigation, signal processing and collaboration with its neighbors;

- Reconfigurability: It should be reconfigurable so that it can fit in various re-

search projects, thus calling for an open hardware and software architecture;

- Robustness: It should be built on a sturdy mobile platform to ensure repeated experiments;

- Easiness of duplication: It should be built mostly out of commercially-off-the-shelf (COTS) parts instead of customized hardware, which will allow other researchers to quickly duplicate the platform for their own research.

In recent years, various existing mobile robot platforms have been developed by different research groups, but most of them can only satisfy a subset of the above criteria. The ASCCbot is created in the Advanced Sensing, Computation & Control Lab (ASCC) at Oklahoma State University. Compared with existing platforms, the proposed ASCCbot is a compact, intelligent mobile robot platform which is relatively inexpensive (around $2500), purely open-source, extendable, duplicable and equipped with basic functionalities. The criteria listed above can be fully satisfied by the ASCCbot.

The hardware system of the ASCCbot is developed based on the iRobot Create and other off-the-shelf components. The software system is built upon the ROS (robot operation system [11]), which includes low-level device control, wireless communication, package management, etc. The inter-robot communication capability of the ASCCbots is facilitated by the networking and distributed computing functionalities of ROS.

The features of the ASCCbot can be summarized as below:

- All components are off-the-shelf, inexpensive, easy to duplicate and extendable;

- It is developed based on ROS, which is open-source and friendly for wireless multi-agent application;

- Basic functions are implemented and robust algorithms are implemented to enable featured functionalities.
- It supports wireless networking for multi-robot research.

1.2 Objective

The objective of this thesis is to develop the ASCCbot, with the concepts of open-source, duplicability, and reconfigurability. To support single or multi-robot research, it is desirable to equip the ASCCbot with some basic and featured functionalities. The ASCCbot's functionalities include integrated navigation, target following, collaborative localization, telepresence with Kinect-based human-robot interaction and semantic mapping. We believe such a robot platform can benefit the robotics research community due to its open hardware and software.

1.3 Outlines

- This chapter presents the motivation and objective of this work;
- Chapter 2 provides the details of the ASCCbot design as well as the implementation of basic functions;
- Chapter 3 presents the function of collaborative localization;
- Chapter 4 illustrates implementation of a telepresence robot based on the ASCCbot;
- Chapter 5 demonstrates semantic mapping through human activity recognition;
- Chapter 6 gives conclusion and future works.

CHAPTER 2

ASCCBOT SETUP AND BASIC FUNCTIONS

Based on all off-the-shelf components, the hardware design of the ASCCbot makes it a duplicable, extendable and robust mobile robot platform. The software of the ASCCbot makes it an opensource platform equipped with fundamental functions and featured functions as well. In this chapter, we first discuss the related work in mobile robot platform development, then the detailed hardware and software design of the ASCCbot is explained. The basic functions which make the ASCCbot a versatile mobile robot research platform are also presented.

2.1 Related Work

We realize that in recent years there exist many multi-robot or multi-sensor platforms developed in various research projects (see in Figure 2.1). However, these platforms do not meet most of the criteria mentioned above. Some of the platforms are developed based on small, tightly resource-constrained simple robots which lack sufficient computational power and sensing capability to conduct accurate navigation or implement desired signal processing and collaboration algorithms. These platforms include the MicaBot [12], CotsBot [13], Robomote [14] robots, and the commercial Khepera robot and its siblings. Some of the platforms are designed from the scratch, which makes it hard to quickly duplicate due to their customized hardware design. These platforms include the COMET testbed developed by Cruz et al. [15], the MARS testbed developed by Grocholsky et al. [16], the Scarab robots [17] developed by Michael et al., and the MVWT-II Hovercraft testbed developed by Jin et al. [18].

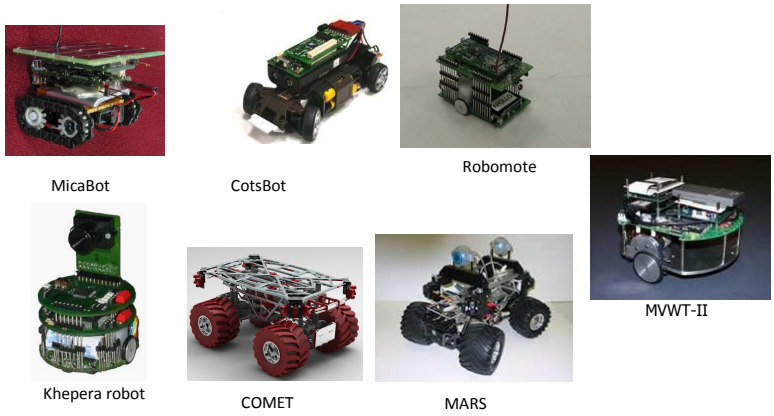


Figure 2.1: Existing mobile robot platforms.

Some of the platforms are developed based on full size commercial robots such as the Pioneer robots, which lack the reconfigurability and may incur significant cost in setting up large sensor networks. These platforms include MIT's multi-vehicle platform [19] and the multiple heterogeneous robot testbed developed under the DARPA SDR program [20]. Additionally, many testbeds [21, 22] have to be constrained in a very small space, which is not sufficient to capture the characteristics of large real world environments.

2.2 Hardware and Software Design

The ASCCbot is built on an iRobot Create with an Atom processor-based computer called FitPC2 [23], a Hokuyo LRF [24] and a Q24 panoramic camera [25]. The iRobot Create is a platform that is designed for robotics development and possesses a serial port through which sensor data can be read and motor commands can be issued using the iRobot Roomba Open Interface protocol. The FitPC2 is a small, light, nettop computer. Hokuyo LRF URG-04LX is a USB-powered device which uses a laser beam to determine the distance to an target. It has a measuring range between 20 mm and 4094 mm, a scanning range of 240°, a scanning rate of 100 ms/scan, a distance accuracy of ±3% and an angle resolution of 0.36°. The fish-

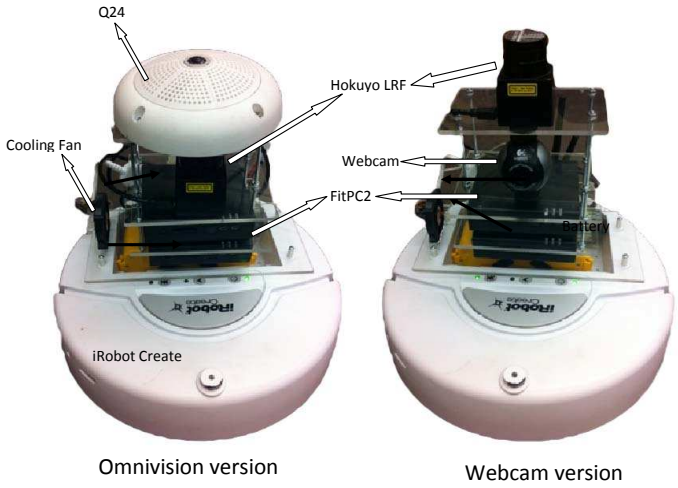


Figure 2.2: Two versions of the ASCCbot.

eye camera (Q24) is capable of providing different views simultaneously including a panoramic view so that it can cover the surrounding area of the mobile platform. The camera provides a highest resolution of 3 Megapixels and color images scalable from 160×120 to 2048×1536, and it uses an Ethernet-based interface. The features of the camera (including resolutions, frame rates, etc.) can be easily adjusted by sending a web request. Moreover, the zooming and panning of the camera lenses can be done by the virtual PTZ function. The camera itself is a web server so that the stream of live images can be obtained by setting up a socket connection. Besides the aforementioned components, external batteries are used to power FitPC2 and Q24 while a usb-powered mini fan is used to cool the FitPC2.

The mechanical design of the platform endows the ASCCbot with stability, and extendability. On top of the iRobot, a plexiglass stand is designed to support the control, communication and sensing devices (see Figure 2.2). The final design of the plexiglass stand has four layers. Devices on the stand are a FitPC2, a Hokuyo laser range finder (LRF), a Mobotix Q24 360°fisheye IP camera, and a USB-powered cooling fan. There are two layout versions for the stand. The first and second layers are reserved for FitPC2s in both versions, in case we need two of them. But for the

Figure 2.3: A multi-robot network.

third and fourth layer, the omnivision version puts the LRF on the third layer and Q24 on the fourth layer; the webcam version puts the webcam on the third layer and the LRF on the fourth layer. The plexiglass stand design fits the iRobot create on the bottom very well, so it is very stable for setting all the components on top of it, and more importantly it can be conveniently extended by adding layers.

ROS (Robot Operating System) is an open-source, meta-operating system for robots. It provides services similar to real operation systems, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. The distributed computing feature of it can also facilitate multi-agent application in a wireless network.

In ROS, a program can be divided into different nodes which can be distributed to different computers in the same network. Nodes are separate processes which can receive and publish messages from and to any other nodes. The driver of one component of the hardware can be treated as a node, while a data processing method can be made as one node as well. As long as all the nodes share one "ROS master".

For the most basic function in the ASCCbot, we have the ROS version driver for iRobot create called $irobot\_create\_2\_1$ from Brown University repository [26] in ROS community. Besides that we also utilized some basic SLAM, navigation, motion, localization packages from ROS repositories. In addition to employing existing works, we designed and implemented some of our own packages for integrated navigation, target following, etc. All these packages or functions will be explained later in this chapter as the basic functions for the ASCCbot.

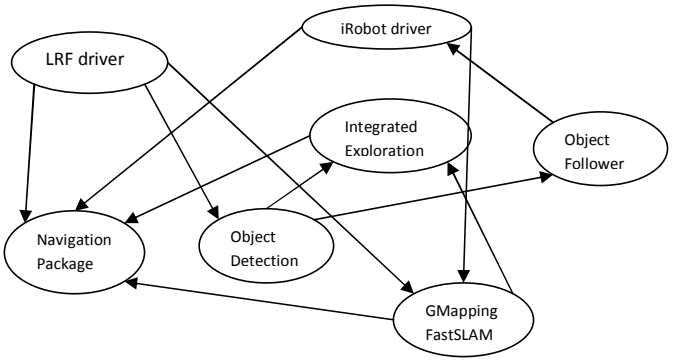


Figure 2.4: ROS nodes structure for the proposed ASCCbot.

The functions we implemented also depend on existing packages. For example, in our integrated navigation package, there are five major nodes (see in Figure 2.4):

- iRobot driver: irobot_create_2_1; (package from Brown University)
- FastSLAM; (GMapping package)
- Semi-autonomous Navigation; (navigation package)
- Autonomous Exploration; (created by the author)
- Target Detection. (created by the author)

Last but not least, as one part of the ROS community, a stack named *ascc-ros-pkg* including $frontier\_detector, target\_follower, semantic\_mapping$, will be released to the whole community as the repository for the ASCCbot. It will be maintained

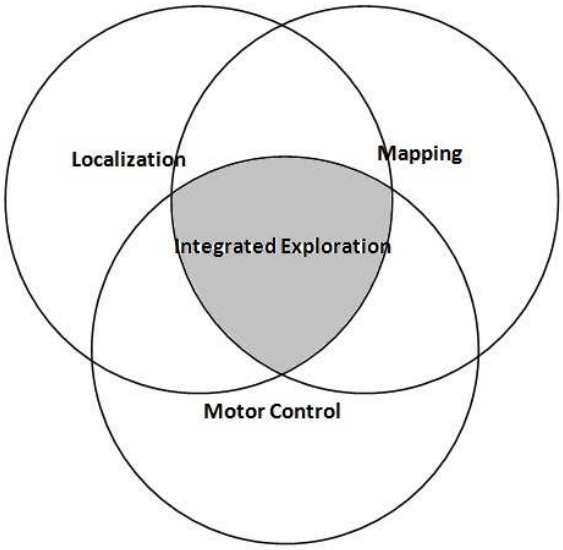


Figure 2.5: Three fields of robotic navigation [1].

and updated by the authors and published to the ROS community as the software resources for the ASCCbot. Anyone who has a ASCCbot or has similar hardware setup as the ASCCbot will be able to use this stack for his own research.

2.3 Integrated Navigation

Localization, mapping and motion control are three partially overlapping fundamental fields of robotic navigation, as illustrated in Figure 2.5. The overlapping portion of all the three fields indicates integrated navigation, which essentially is the strategy to guide the mobile robot to move to an unknown environment and map it [1].

Various researchers have emphasized the importance of integrated navigation and proposed different approaches to it. The common approach is to generate potential candidate goal points, and then use a utility function to associate all the three parts and evaluate the contribution of all the candidates based on the utility function. The selected candidate goal point should have a maximum value from the utility function. The robot will then be commanded to go to the selected goal points until it arrives and then it will repeat the progress above.

Candidate Goal Point Generation

In order to generate candidate goal points on the partially created map, a frontier-based approach is adopted [27]. Conveniently, the 2D grid occupacy map created from GMapping package uses only 3 types of grids which have value -1, 0 and 100 to indicate unexplored space, free space and obstacle respectively. Based on the format of the map information, the frontier-based candidate generation process is divided into three phases. In the first phase (Preliminary Generation), the frontier candidates which have value 0 and a certain neighborhood of grids will be extracted. But it will give too many candidates. So in the second phase (Filtering), a filter is necessary to filter out some obviously impossible candidates. After filtering, the final phase (Grouping) is important for clustering the points. The outcome of the candidate generation process should be well-separated candidate goal points.

Some results of the candidate goal point generation process for an office room and the corridor are shown in Figure 2.6. Dark stars and circles indicate the results after the filtering and the grouping phases respectively.

Utility Function Construction

An utility function is used to select one final frontier goal point out of all the candidates. What makes one candidate stand out from others depends on both the position of the candidate and the robot platform. Based on the position of the candidate, the information gain and localizability can be evaluated. Information gain means how many unexplored grids can be seen from the candidate position while localizability depends on the number of landmarks visible from there which indicates how easy the robot can relocalize itself. From the position of the robot, the estimate of the position certainty is given from Rao-Blackwellized particle filter [28] from FastSLAM GMapping node. The entropy here affects the evaluation of the traveling distance from the robot position to the candidate position. In the utility function, a shorter path is more appealing and its effect will be influenced by the

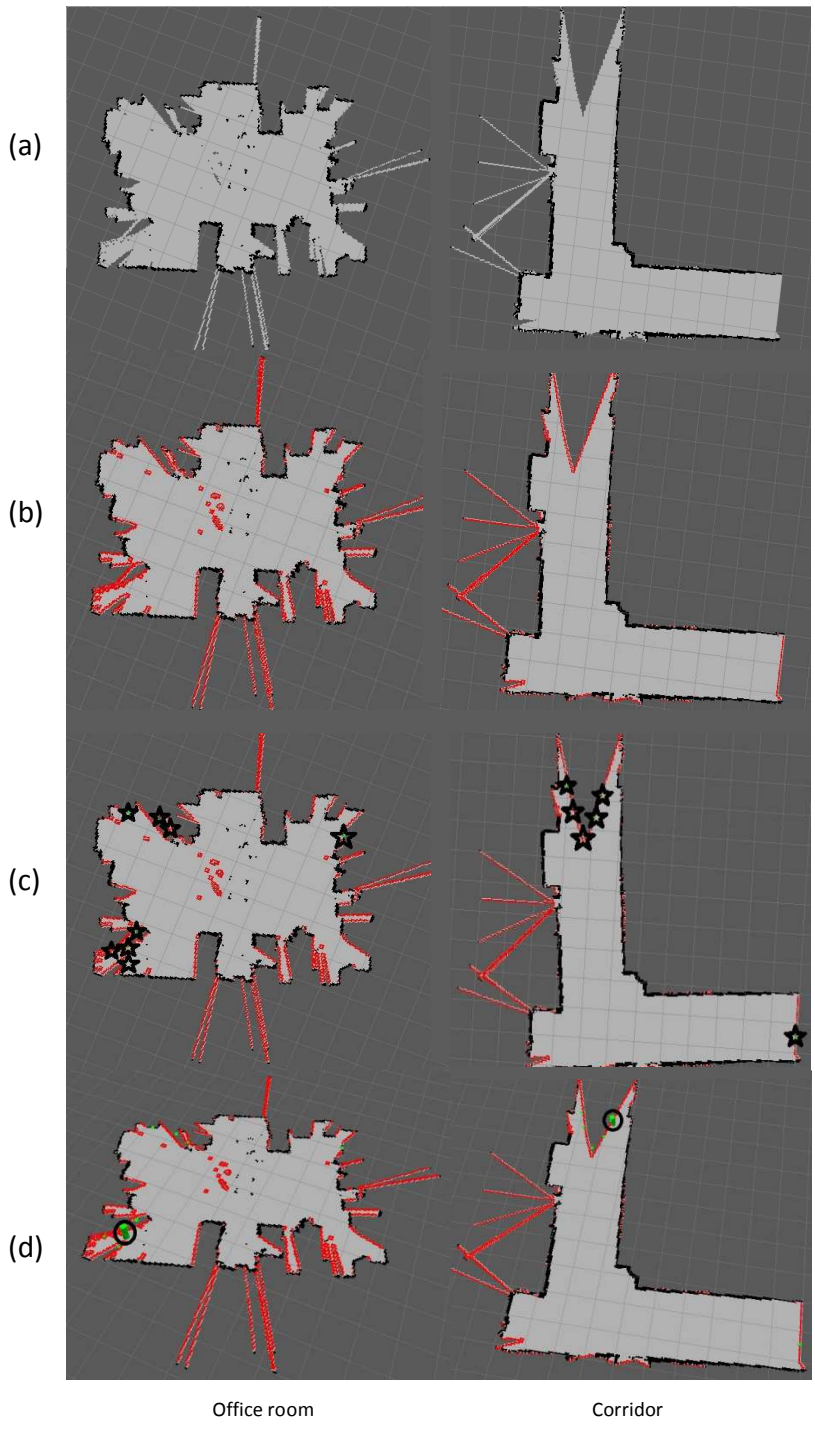


Figure 2.6: Frontier generation process for an office room and an corridor: (a) Raw map; (b) Frontier candidates after preliminary generation; (c) Frontier candidates after filtering; (d) Frontier candidates grouping.

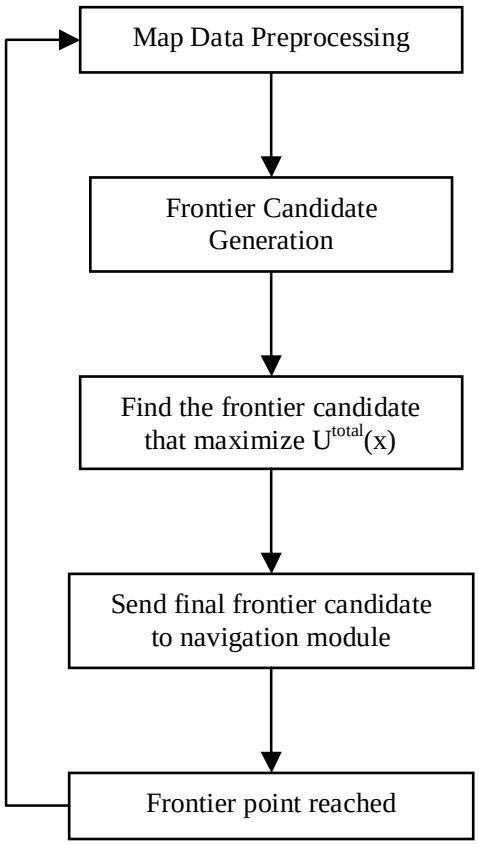


Figure 2.7: Flow chart for integrated navigation.

entropy of the robot position.

$$U_{total}(x) = w_S w_I U_I(x) + w_E w_N U_N(x) + w_L U_L(x) \tag{2.1}$$

In Equation (2.1), w_I, w_N and w_L are the relative weights for information gain, navigation cost and localizability respectively. w_S denotes the scenario coefficient, for example, office rooms and corridors have different w_S values, while w_E indicates the entropy's effect on the navigation cost. The goal position is the one that maximizes the utility function.

$$x^* = \arg\max_x(U_{total}) \tag{2.2}$$

The flowchart for integrated navigation is shown in Figure 2.7.

2.4 Target Detection and Target Following

The task of target detection is to recognize certain target and to locate it when the autonomous robot is exploring an unknown environment. Compared to normal surveillance systems, our camera is mounted on a moving platform so the light condition is changing and the images contain a significant amount of noise.

Considering the computation ability of our platform, we design two different levels of the image processing to detect objects with different computational complexities. The low-level is to detect an orange traffic cone simply using color segmentation which is fast and with low computation, while the high-level is to recognize the OSU logo by taking advantage of SURF (Speeded Up Robust Features) [29] features which is more time consuming but also more effective.

Algorithm I: Cone detection

A color segmentation technique is used to detect the traffic cone. The purpose of color segmentation is to find contiguous regions in which individual pixels share common characteristic. It is a simple and fast way to detect the traffic cone due to its distinctive color. Since the RGB color space is very sensitive to lighting changes, the captured image is converted from the RGB color space to the HSV color space first. The HSV space decouples the intensity component from color carrying information. Then color segmentation in the HSV color space can be achieved by setting a threshold for each channel. A binary image is obtained by filtering HSV channels. After applying a Gaussian filter and a morphology method (dilate and erode) to reduce the noise, the cone region is detected just as shown in Figure 2.8.

Algorithm II: Logo recognition

Sometimes the target we want to detect may not be as easy as traffic cones, for example, logos or trade markers. Therefore, local features should be used to provide detail information about the target. Here we use SURF features to detect the logo "Pete" of Oklahoma State University. Firstly SURF features are extracted

Figure 2.8: Detection using color segmentation.

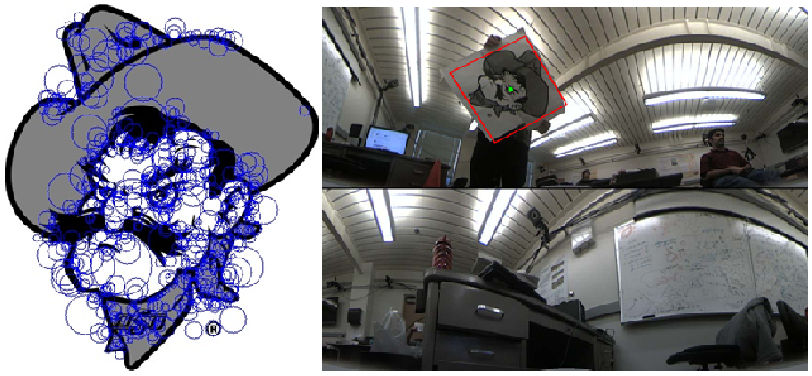


Figure 2.9: OSU logo recognition using SURF features.

from each new image frame. Then feature matching is accomplished by computing the correspondences between the image features and the target template in our database. Based on the matched features the target is detected by an affine transformation with the target stored in the database just as shown in Figure 2.9.

Based on the target detection algorithm and the GMapping package, the ASCCbot can follow the moving target around the environment and create the map of the surroundings simultaneously. The node of target detector provides the bearing of the target from the robot point of view, while the LRF can find the target distance between the ASCCbot and the target at a specific angle. Then a corresponding control strategy is applied to control the ASCCbot to move toward the target.

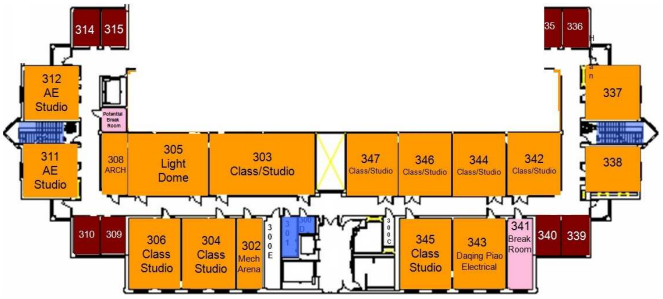
Figure 2.10: Navigation at a corner of the corridor.

2.5 Performance Evaluation

The performance of the basic functions of the ASCCbot is evaluated through different experiments. During the experiments, some general parameters of the ASCCbot can be observed: the battery of the platform can last more than 3.5 hours and the wireless communication distance is more than 100 meters in an non-cluttered environment..

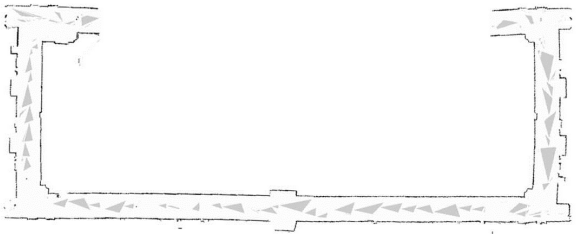
2.5.1 Integrated Exploration

The performance of the integrated exploration is evaluated by running the ASCCbot at the corner of a corridor. The trajectory in Figure 2.10 shows how the ASCCbot is given a sequence of frontier goal points and navigates to them one after one. The frontier goal points are generated from the integrated exploration algorithm. The growth of the map is not shown in Figure 2.10, but the footprints of the ASCCbot represent the frontier points of the corresponding map, which can illustrate the growth of the map boundaries. The performance of integrated exploration shows that the ASCCbot is able to continuously navigate to the unexplored area of a corridor corner.



ATRC building 3rd floor partial floor plan

Pioneer map



ASCCbot map

Figure 2.11: Corridor map comparison.

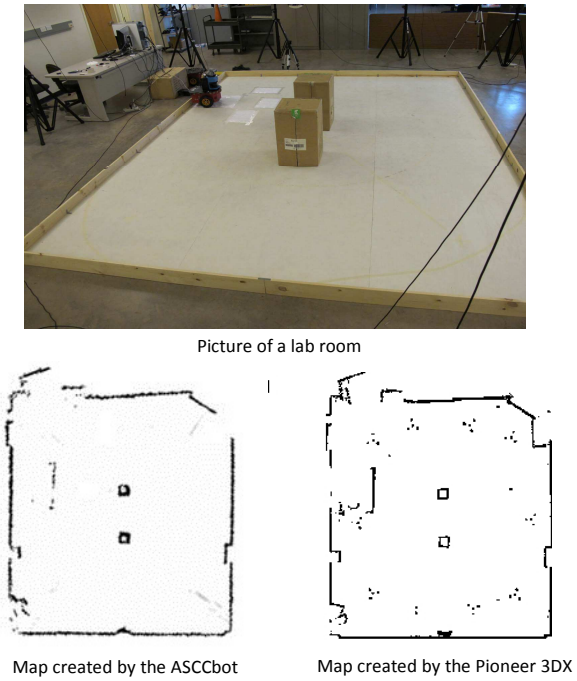


Figure 2.12: Lab room map comparison.

2.5.2 Corridor Mapping & Room Mapping

Both the ASCCbot and the Pioneer 3DX [30] are used to create a partial corridor map of a floor in a building. On the Pioneer robot, a Sick laser [31] and wheel encoders are used. The long side of the corridor is about 100m while the short side is about 25m. It took the ASCCbot about 20 minutes to complete the task at the speed of 0.3 m/s. The details of the map including the concave at the middle of the long side, the corners and the benches at the short side are clearly shown in the map. Most importantly, the length of the sides and the shape as a whole is very close the the floor plan. It took the Pioneer 3DX around 18 minutes to create the map. The overall shape and the map details in the Pioneer map are very close to the one obtained by the ASCCbot. However, the length of the corridor in the ASCCbot map is shorter than that in the Pioneer map, and the corner angles are also slightly different in the two maps. These differences are due to the error of the encoder and the laser of the ASCCbot.

A lab room is also mapped by both the ASCCbot and the Pioneer 3DX (see Figure 2.12). It is a relatively small room with one table and two boxes in the middle. The details at the boundaries, the table and two boxes are very close in the two maps. But at the upper right corner of the left map, some features are missing. This is probably due to the materials that reflect the laser beam from the Hokuyo LRF while not for the SICK laser on the Pioneer robot. The groups of three dots at the middle indicates the tripods, which are recognized in the Pioneer map but not in the ASCCbot map. It is because they are filtered out in the GMapping algorithm of the ASCCbot.

2.5.3 Target Detection

Moving target detection for mobile robot has been studied by many researchers [32] [33]. In the target following scenario, the rotation of the robot affects the target detection performance most. In order to achieve an acceptable target detection accuracy, the rotation rate has to be limited. Experiments were designed to evaluate the relation between the rotation rate and the objection detection accuracy as described below.

With an image resolution of 1024×768 pixels the processing speed is around 4fps for cone detection and 2fps for logo recognition which ensures the real-time processing. The target (cone or logo) is deployed 1 meter away from the mobile robot which is rotating at different levels of speed (0.26 rad/s, 0.52 rad/s and 0.87 rad/s). For each speed we ran our program 5 minutes. The experiment results are shown in Table 2.1. The images become more blurred when the speed of rotation increases, which affects the accuracy of target detection (see Figure 2.13). From Table 2.1, it is shown that for the low-level recognition the detection rate drops only a little with changing of speed which means that color features are quite robust. For the high-level recognition the detection rate drops significantly with blurred images which means the rotation

rate should not be too fast if local features are used.

Table 2.1: Detection rate with different rotation speed.

Detection Accuracy / Rotation Speed / Detection Mode	Low speed	Medium speed	High speed
Cone Detection	99.09%	97.41%	97.05%
Logo Recognition	91.89%	64.15%	25.86%

Figure 2.13: Target detection performance for cone detection(left column) and logo detection(right column) with different rotation speed ((a) Low speed (b) Medium speed (c) High speed).

CHAPTER 3

COLLABORATIVE LOCALIZATION

Target localization is one important part of a mobile robot network. In order to provide better localization results, we believe the fusion of the localization results from multiple robots would improve the localization performance. With the object detection and networking feature of the ASCCbot, it is possible to implement a collaborative localization function as mentioned. In this chapter, the details of the collaborative localization function will be described.

3.1 Introduction

Mobile robot networks have the ability to collect environmental information using a set of mobile robots. With features such as wireless communication, distributed computing, and mobility, a mobile robot network has great potential in many applications such as monitoring and surveillance. In order to support the research in networked surveillance and prove the capability of the ASCCbot, we designed and implemented the collaborative localization function.

Target localization is one important step for surveillance. A collaborative target localization experiment is designed and carried out. The idea is that each single ASCCbot can detect the target and give an estimate of the target location. Then different target location estimates from multiple ASCCbots will be fused by considering the uncertainty of each estimate. At last, the location estimate after fusion will be compared with the ground truth provided by the Vicon motion capture system [34].

3.2 Network Setup of Multi-robot System

The ASCCbot is equipped with the capability to extend to a multi-robot platform. Three ASCCbots are configured for collaborative localization. They have exactly the same hardware setup. We name them robotA, robotB and robotC respectively. The FitPC2 connected to each of them has a different IP address by which we can differentiate them. The software system of the multi-robot ASCCbot platform is also built on ROS. The networking feature is built upon the ROS network functions.

In the collaborative localization, we utilize functions from existing ROS packages such as: localization, SLAM, and the self-developed packages such as $target_follower$. Besides these, the networking feature of ROS is extensively utilized. The inherited distributed computing feature of ROS makes the networking of multiple robots straightforward. Any ROS node can run in one computer or be distributed to multiple computers, and *roscore* can also be put on any computer within the network. For each single computer, ROS_MASTER_URI and $ROS\_IP/ROS\_HOSTNAME$ are used to define the *roscore* IP and its own IP. Once the IP addresses are set up, all the computers can communicate with each other. If multiple computers want to run the same node, *namespace* need to be used. In our case, *namespace* $robotA/, robotB/, robotC/$ are used to specify different mobile sensors.

There are two types of setup for the network. One is a distributed setup (see Figure 3.2) where all the nodes are equal. The other one is a centralized setup (see Figure 3.1) where there is a laptop or other portable computer ("workstation" in Figure 3.1) which is responsible for the networking.

3.3 Target Detection on the ASCCbot

In the collaborative localization, the target detection algorithm will be carried out on three ASCCbot. The target to be localized is detected by the same algorithm used

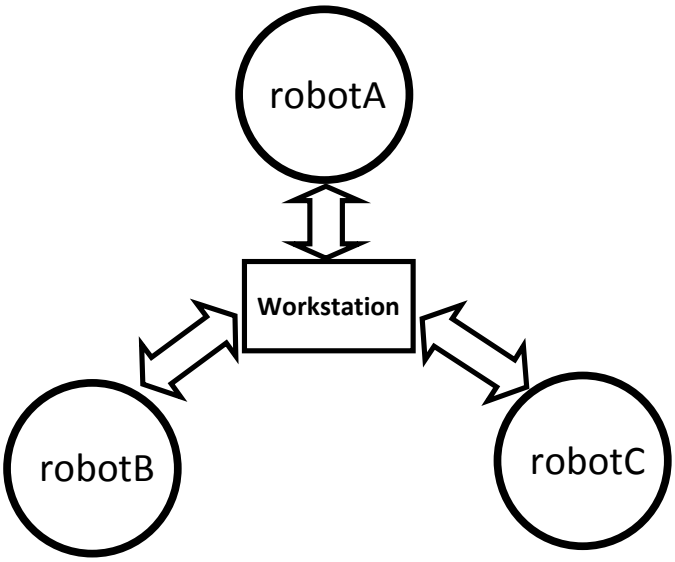


Figure 3.1: Centralized network setup

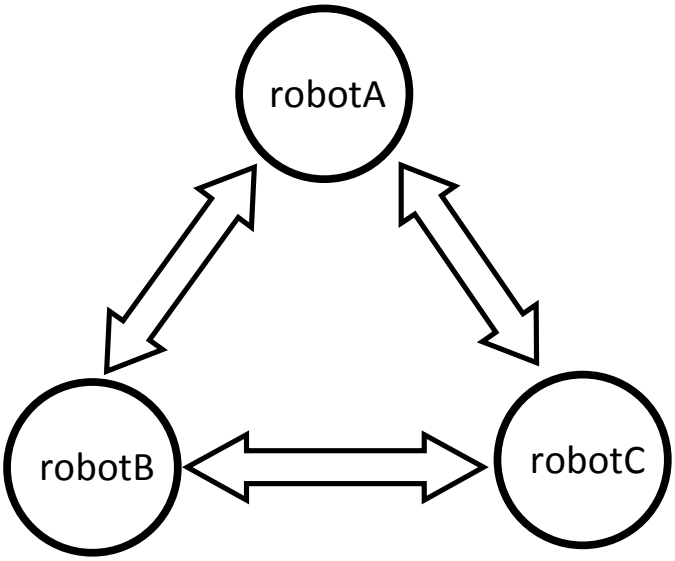


Figure 3.2: Distributed network setup

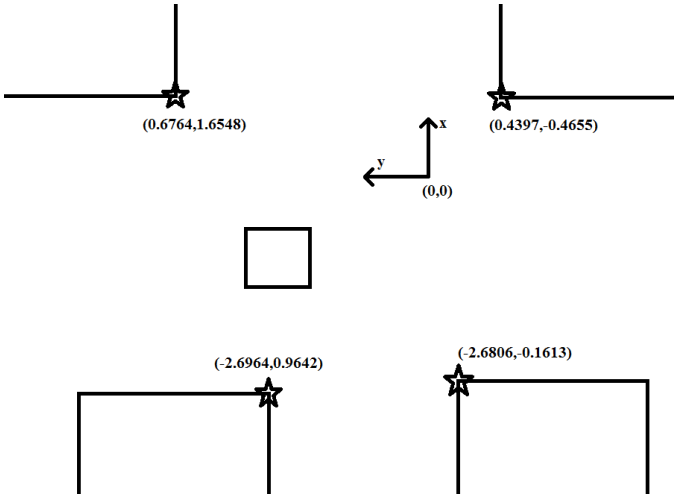


Figure 3.3: Landmarks in the Vicon coordinate

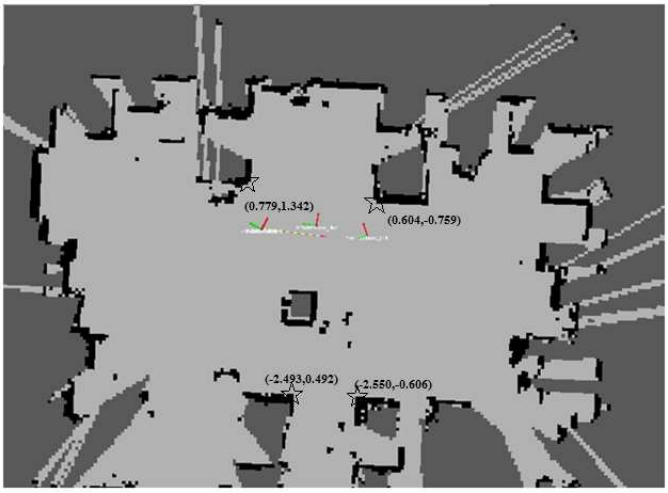


Figure 3.4: Landmarks in the ASCCbot coordinates

in the function of target following. In order to achieve a reliable target detection performance on a multi-robot platform, the low-level target detection algorithm as described before is used. In the designed experiment for collaborative localization, an orange traffic cone was used as the target to be detected with three ASCCbots.

3.4 Collaborative Target Localization

3.4.1 Calibration

In order to compare the target location estimate from the the multiple robots with target location ground truth from the Vicon motion capture system, a calibration process needs to be carried out to align the coordinates of the Vicon system with those of the mobile robots. Four desk corners are used as landmarks for the calibration. Figure 3.3 and Figure 3.4 show the map from the Vicon system and the mobile robots, respectively.

3.4.2 Collaborative Localization

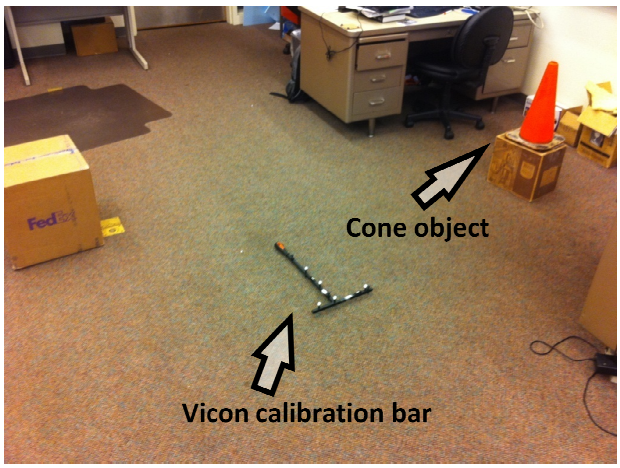


Figure 3.5: Vicon calibration.

After the calibration process, the collaborative target localization experiment can proceed. First, the map of an office room is created by running Simultaneous Localization and Mapping (SLAM) [35] on one of the ASCCbots. The starting pose of the ASCCbot will determine the origin and the orientation of the map. Three ASCCbots will be placed inside the office room with AMCL node running, which will publish the robot position estimates through Monte Carlo localization algorithm [36] (see Figure 3.6). Meanwhile, the Q24 camera will detect the target and output the angle with

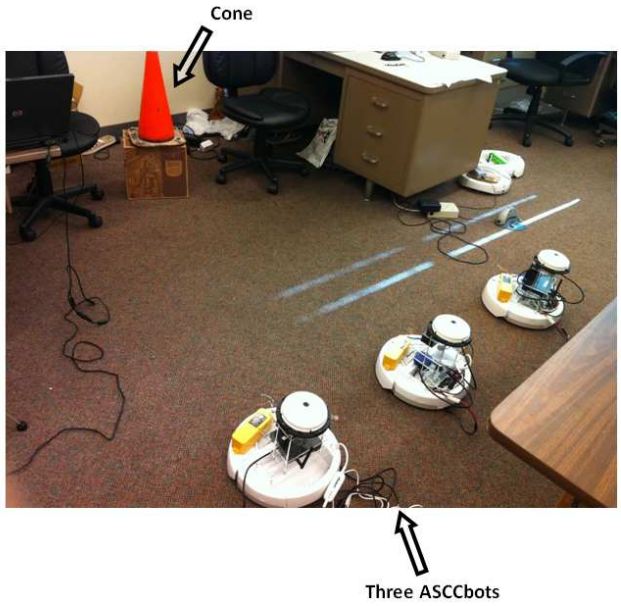


Figure 3.6: Collaborative target localization with three ASCCbots.

respect to the ASCCbot. With the help of the angle, the laser can read the distance at that angle, obtaining the distance estimates to the target. After the robot pose, the target angle and target distance estimates are all acquired, the target location estimate can be computed according to (3.6) and (3.7). Since the pose estimates from Monte Carlo localization has uncertainty, the three target location estimates can be fused based on those uncertainties. The ground truth from the Vicon system will then be compared with the fused estimation result.

The fusion strategy is summarized as the equation below:

$$W_a = \frac{1/U_a}{1/U_a + 1/U_b + 1/U_c} \tag{3.1}$$

$$W_b = \frac{1/U_b}{1/U_a + 1/U_b + 1/U_c} \tag{3.2}$$

$$W_c = \frac{1/U_c}{1/U_a + 1/U_b + 1/U_c} \tag{3.3}$$

$$X_{fusion} = X_a \cdot W_a + X_b \cdot W_b + X_c \cdot W_c \tag{3.4}$$

$$Y_{fusion} = Y_a \cdot W_a + Y_b \cdot W_b + Y_c \cdot W_c \tag{3.5}$$

where, W stands for the weight, while U indicates the uncertainty.

$$X_{estimate} = x_{robot} + distance \times \cos\left(angle_{robot} + angle_{target}\right) \tag{3.6}$$

$$Y_{estimate} = y_{robot} + distance \times \sin\left(angle_{robot} + angle_{target}\right) \tag{3.7}$$

3.5 Performance Evaluation

The performance of collaborative localization is evaluated in this section. Three sets of experiments were carried out. In each set, three robots all had their own estimates and the uncertainty of their own pose estimates, which come from the Monte Carlo localization algorithm. After applying the equations described in the last section, the final target location estimates are given. The results show that the fusion strategy does improve the target location estimate with the help of three robots. The estimation results in the 2D map are shown in Figure 3.7, while the numerical data of all three sets of experiments are shown in Table 3.1 and 3.2. From the data in Table 3.1 and 3.2, we can see that the x coordinate estimate after fusion is closer to the the ground truth x coordinate than every single estimate from each robot. The difference between the data after fusion and the ground truth data is less then 4 millimeters which is acceptable compared the size of the environment.

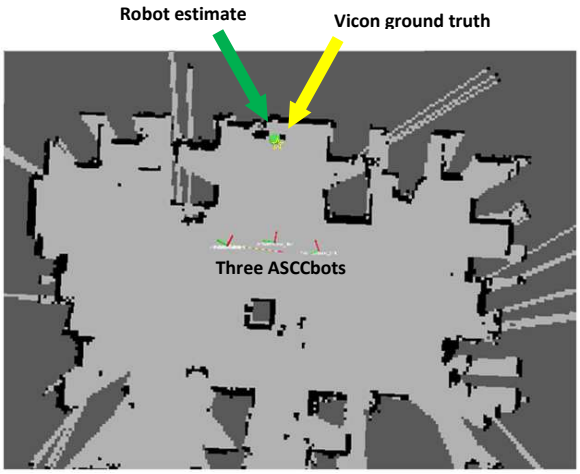


Figure 3.7: Collaborative localization result.

Table 3.1: Target location estimates from ASCCbots. The unit of x and y coordinate are meters while the unit for uncertainty is square meter

	X estimation			Y estimation			Uncertainty					
	Set1	Set2	Set3	Set1	Set2	Set3	Set1 x	Set1 y	Set2 x	Set2 y	Set3 x	Set3 y
RobotA	0.77325	1.67626	1.39862	0.22630	0.70589	1.2412	0.10967	0.000935	0.01248	0.002561	0.01476	0.00780
RobotB	0.78929	1.52608	1.3822	0.22650	0.6998	1.2602	0.0060329	0.002237	0.012039	0.0012959	0.017079	0.002451
RobotC	0.81590	1.51796	1.4641	0.26530	0.7162	1.286	0.008578	0.0015116	0.009264	0.000376	0.0225	0.003677

Table 3.2: Comparison between the estimate after fusion and the ground truth. (unit is meter)

	X estimation after fusion	X from vicon	Difference X	Y estimation after fusion	Y from vicon	Difference Y
Set 1	0.79946	0.7621	0.03736	0.25819	0.2256	0.03259
Set 2	1.56716	1.596	-0.02884	0.71187	0.7413	-0.02943
Set 3	1.41002	1.455	-0.04498	1.26587	1.266	-0.0013

CHAPTER 4

TELEPRESENCE ROBOT

Telepresence robots can allow a user to interact with remote people through video and audio communication. In this chapter, the implementation of a realworld telepresence robot based upon the ASCCbot is presented.

4.1 Introduction

The term telepresence represents a relatively new direction of robotics application with the advancement of the Internet technology. It refers to a set of technologies which allow a person to feel as if they were present, to give the appearance of being present, or to have an effect, via telerobotics, at a place other than their true location. There are already some commercially available telepresence robot in the market, such as Texai from Willow Garage, QB from Anybots, VGo roobt, etc (see in Figure 4.1).

4.2 Hardware Setup

The telepresence robot can be divided into two parts: the robot end and the remote end. The ASCCbot is the robot end. In order to display the remote end video at the robot end, we put an iPad on the robot (see Figure 4.2). There is an additional stand designed and put onto the original ASCCbot. It does not block other sensors which are important for running other functions. We also add a Kinect sensor on top of the robot. The Kinect sensor is used to make the robot controllable by the people around it. That means, the robot can be controlled by both the remote end user and the people around it. With the entire stand, the Kinect and the iPad, the ASCCbot

Figure 4.1: Commercial Telerobots.

can run very stable. The remote end is a user interface software, which may include a joystick.

4.3 Software Setup

In order to make an intuitive interface on the remote side. A GUI (Graphic User Interface) written in QT [37] is created in the ROS environment. By using the GUI, we can remotely control the ASCCbot as long as we input the IP address of the FitPC2 that is connected to it and we are also able to observe the video streamed from the robot end to the GUI wirelessly. The video data is compressed in the JPEG format, and decompressed at the remote end for the GUI rendering. Another interesting feature in the GUI is the 2D metric map created by the SLAM program which can be shown in the GUI continuously in realtime. This feature enables the remote user to observe the map created by the ASCCbot and have a better understanding of the surrounding environment.

At the robot end, a hand gesture recognition algorithm is implemented for the

Figure 4.2: ASCCbot telepresence platform.

people around the ASCCbot to control it. Imagining the following scenarios: the remote end user somehow loses control of the robot and it is still running; the remote user is not realizing the robot is too close or too far away from the people he wants to talk to; or the remote user wants the people around the robot to lead him to observe the remote environment. In the above scenarios, the Kinect-based gesture control at the remote end will be very useful.

There are two motion commands from the robot end which are "come close" and "go away", based on which the people around the robot would be able to control the robot using their gestures. The details of the Kinect-based gesture recognition for the ASCCbot control will be presented in the following section.

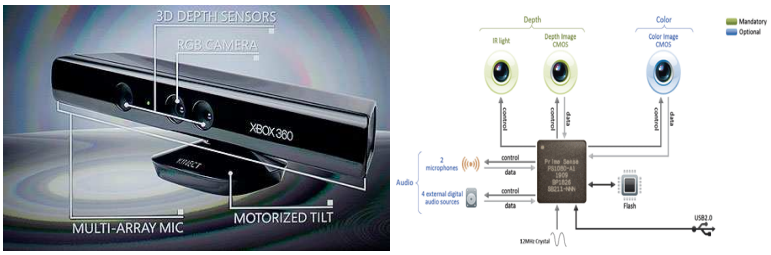


Figure 4.3: XBox 360 Kinect

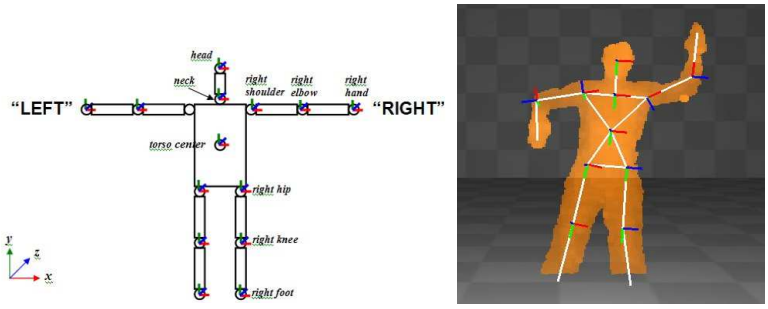


Figure 4.4: Skeleton model

4.4 Kinect-Based Gesture Recognition for ASCCbot Control

We would like to develop a robust and intuitive strategy for human to control the ASCCbot through simple gestures. Thanks to the recent release of XBox 360 Kinect sensor [38] and *openni_tracker* package from ROS, we are able to extract a skeleton model of a human subject standing in front of the Kinect sensor. The Kinect sensor is a webcam-style add-on peripheral for the Xbox 360 console (A gaming device of Microsoft). It is one of the most popular electronic products for entertainment since it was released in November, 2010. This product has integrated several techniques including 3-D imaging, audio processing and motor control, which is shown as Figure 4.3. A ROS-based driver *openni_tracker* is applied to obtain both the raw data of the 3D scene and the skeleton model information as in Figure 4.4.

It is an intuitive way to use 2-D or 3-D model to describe the structure of human body. A human skeleton model is used and represented by the "joints" as shown in Figure 4.4. The whole model contains 15 joints for different human parts such as

head, torso, hand, etc. Each joint has its position and orientation. Orientation is calculated by two or more joint positions. Additionally, each joint has its confidence from 0 to 1 which indicates the detection accuracy. The first step is to obtain the position and orientation data. After the data are obtained and segmented, pattern recognition techniques can be directly applied to identify human gestures.

In machine learning, pattern recognition is the assignment of some sort of output value (or label) to a given input value (or instance), according to some specific algorithm. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes (for example, determining whether a given email is "spam" or "ham").

Based on the repetitive feature of the human movements, template matching could be an effective method. In the template matching approach, a typical template of the movement need to be chosen. And the realtime input sensory data will be sampled by a fixed-size time window and compared with the template data. The template data size is the same as the window size. Each point of the sample data will be compared with the template accordingly. And the sampled data will be shifted along with the window, in order to find the least mean square error. If the least mean square error is smaller than a pre-defined threshold, then the sample window will be claimed to be similar to the template. The template data, testing data and comparison result are showed in Figure 4.5.

In our work, only the yaw and pitch angle of the right elbow are considered. There are two pre-obtained templates for both the yaw and pitch angle of the right elbow. At the robot end, the skeleton model of the people standing in front of the Kinect sensor will be obtained. The yaw angle and pitch angle from the skeleton model are fed into the template matching program continuously. If one template is matched with the realtime data, an signal will be emitted to the ASCCbot motion control node. There are two types of signal which show the matching of two templates and

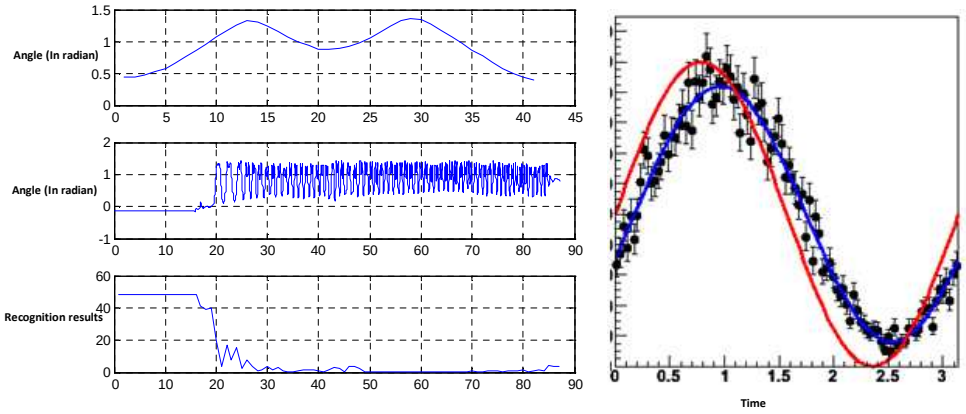


Figure 4.5: Convolutional template matching

represent the two commands: "come close" and "go away". The overall system setup can be found in Figure 4.6.

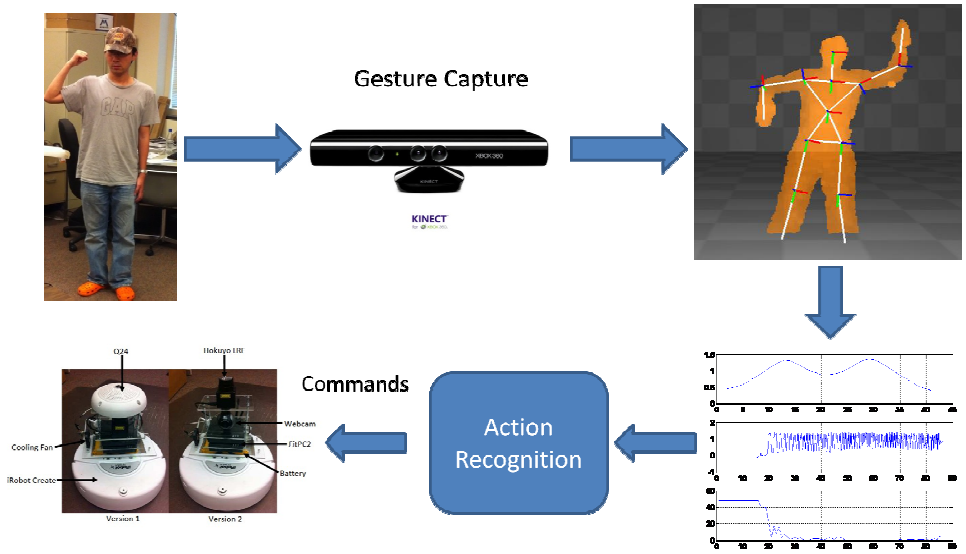


Figure 4.6: Overall system setup

4.5 Comparison with Existing Telepresence Robots

Compared to the commercially available telepresence robot, the ASCCbot telepresence platform is equipped with a Kinect sensor which allows the robot to be controlled by nearby people. The scenario can be pictured like this: the remote user accidentally drive the robot too close to the people he want to talk with, so that people

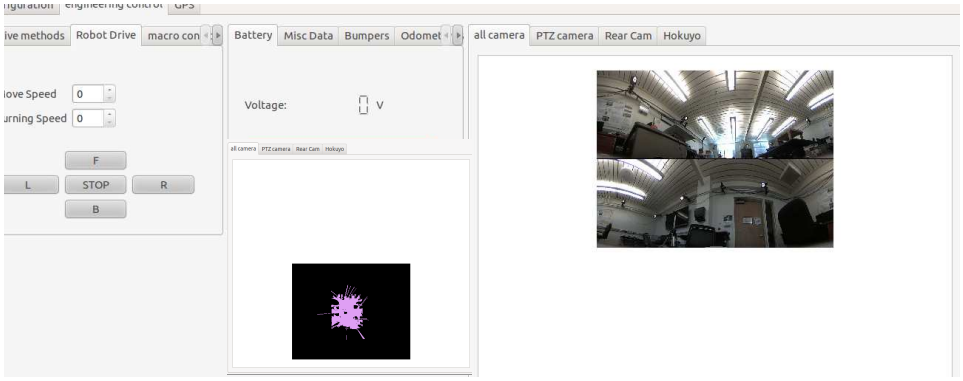


Figure 4.7: The QT GUI for teleprsence

can be able to push the robot backward by using some intuitive gestures in front of the Kinect sensor. The other scenario is, if the robot is far away from the people you want to have conversation with, that people can ask the robot to come nearer. We recognize simple human gestures by monitoring the right elbow yaw/pitch angles from the human subject in front of the Kinect sensor, which can be extracted from the Kinect skeleton model.

Beside gesture control through the Kinect sensor, we are also able to render the 2D metric map created from the underlying SLAM algorithm (see Figure 4.7). While the robot is wondering around the indoor environment either by the control from the remote user or the people nearby, the 2D map of the environment is being created and shown to the remote user. The future function to be realized is that, the remote user can click on the 2D map to select the goal point he wants the telepresence robot to reach.

CHAPTER 5

SEMANTIC MAPPING THROUGH HUMAN ACTIVITY RECOGNITION

In recent years, people are trying to endow mobile robots with complex functions such as action planning, recording and recalling episodic memories, reasoning about spatial concepts and their relations, etc. Researchers started to realize the importance of semantic information for the complex high-level functions. The process of creating a semantic map is to obtain a representation of the environment which grounds human concepts to instances of spatial entities. Various approaches were adopted to obtain semantic information such as: 2D image processing, geometrical laser data processing, 3D point cloud processing, voice recognition, etc. In the ASCCbot, we propose to use human activity recognition to extract the semantic information, which is more intrinsic. The activity recognition part is realized by using wireless motion sensors developed in ASCC lab. In this chapter, we will first review the related work in semantic mapping and introduce the overview of our approach. Then we provide the details of the overall system setup and wearable sensor-based human activity recognition. At last, both the simulated and real world experiments are presented.

5.1 Related Work in Semantic Mapping

The importance of including semantic information in robot maps has been recognized for a long time [39, 40]. In recent years, several researchers have been trying to create a system in which the robot can acquire and use semantic information [41]. Some of them tried to manually obtain semantic information via a linguistic interaction with

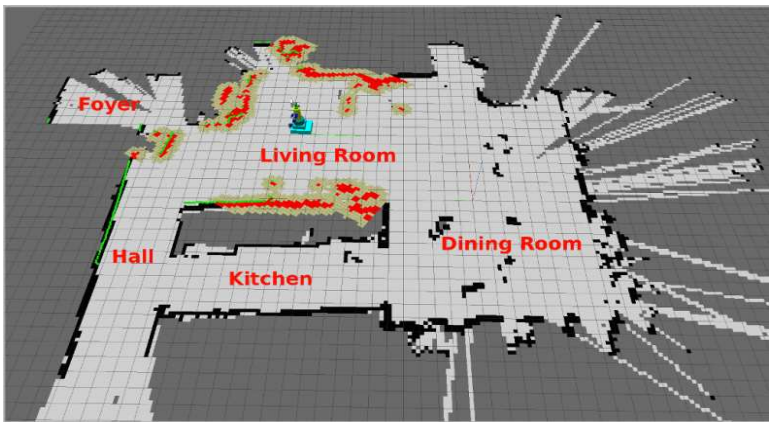


Figure 5.1: A 2D semantic map as an example

a human. Some of them were limited to the classification of surface elements [42] like ceilings, floors, etc. In many other works, people are trying to acquire semantic information through the recognition of 3D models. In [43] and [44] specific places of indoor environments are labeled based on the presence of key objects in them through computer vision techniques that extract the necessary information from images. Nuchter *et al.* tried to extract semantic information from 3D models built from a laser scanner on the robot [45]. Nielsen *et al.* used a snapshot technique to mix 2D images onto 3D objects [46]. Jebari *et al.* utilized panoramic cameras to extract high-level information using object recognition [47]. All of these vision-based methods suffer from high computational demand, difficult segmentation and high background noise. Other researchers focused on the theory of establishing multi-hierarchical semantic map creation process like in [48], [49] and [50]. The approach we use can provide a more efficient and more essential way to acquire semantic information through the interaction between human and objects.

Figure 5.2: The setup of hardware system in a mock apartment.

5.2 Semantic Mapping Through Human Activity Recognition

5.2.1 Problem Statement

The considered scenario is an unknown human-robot coexisting indoor environment with continuous human-robot interaction. The overall concept of semantic mapping through activity recognition is illustrated in Figure 5.2.

Initially, the robot enters an unknown environment. Let R denote the considered room types, $R = 1, \cdots, N$. For different room types, different sets of furniture may be involved and may have different deployment in the room. The indoor area is divided into grids. Assume that the total number of grids is k. Any location within the area is mapped into a grid index through the following function: $G = g([X, Y])$, where $G \in \{1, \cdots, k\}$ and $[X, Y]$ is the coordinates of the location. To build a semantic map of the room, the robot needs to not only detect the surrounding objects and their locations, but also label the objects with the corresponding furniture type. Since we have no prior knowledge of the room, we assume $P_0(R) = \frac{1}{N}, P_{G,0}(F) = \frac{1}{M}$, where G is the grid where an target is located. F denotes the furniture type, $F = 1, \cdots, M$. The uniform distributions are assumed at the initial stage to denote the least prior

knowledge, i.e., the entropy is the largest. After the map is built, we expect to have a more informative probabilistic distribution $P_{G,t}(F)$, based on which a more informative distribution $P_t(R)$ can be obtained. Here, t denotes the time index.

The robot can detect surrounding objects and their locations, based on which its own location and orientation can be determined and is denoted by $L_{r_t} = [x_{r_t}, y_{r_t}, \theta_{r_t}]$. Without complicated vision data processing, however, the robot cannot distinguish one object from another, which inevitably incurs the data association problem in some situations. The problem can be more severe when the environment is crowded with detectable objects or there is large measurement noise. The obtained metric map could be erroneous.

In our framework, there is another source of information, i.e., the activity of the human subject. Generally, the human subject performs certain activities around certain furniture. For example, "sit" on a "chair", "lie" on a "bed". By recognizing the human activity, the knowledge about the furniture type at the human's location can be learned. Furthermore, the location can also put a constraint on possible furniture type. For example, a cabinet is more likely to be placed against a wall or in a corner, while a dining table is usually away from walls. The goal of this work is to incorporate this information for semantic and metric map creation. The robot is able to perform higher level tasks based on the the semantic information.

5.2.2 Sensor Fusion for Activity-based Semantic Mapping

In this section, we discuss the sensor fusion approach to creating the semantic map. The overall architecture of the system can be illustrated by the diagram shown in Figure 5.3.

Human Activity Channel

At each time t, the human subject performs certain activity, e.g., sitting, lying. Let A_t denote the true activity type at t and O_t denote the corresponding estimated

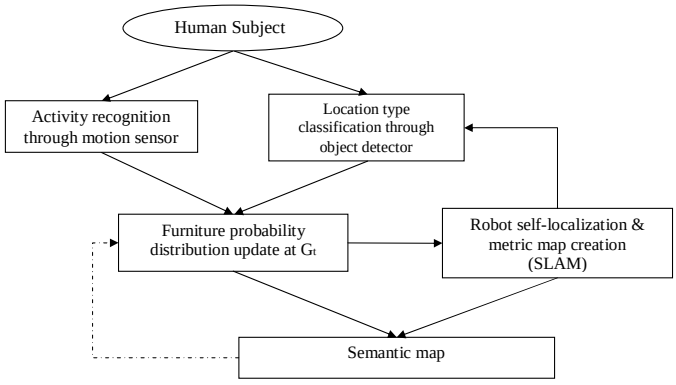


Figure 5.3: The system diagram of the sequential semantic map learning method.

activity based on the measurement of the motion sensors. The observation model is $P(O_t|A_t)$, which gives the likelihood of O_t when the activity A_t is known. Since this is the characteristic of the motion sensor, it is generally time-invariant and can be obtained from supervised learning. In this work, a neural network based method [51] is proposed for activity recognition.

On the other hand, the activity is associated with the furniture type through the model $P(A_t|F)$. For example, when the furniture "bed" is given, the probability of "lying" and "sitting" are much higher than that of "standing". This knowledge can be obtained empirically.

Based on the rule of total probability, we have:

$$P(O_t|F) = \sum_A P(O_t|A)P(A|F) \tag{5.1}$$

We obtain the activity observation model when the furniture type is given.

Human Location Channel

The target detector on the robot can detect and localize the human subject. Let d_t and α_t denote the range and bearing of the human subject with respect to the robot at time t. Then the location of the human subject can be found as $L_{h_t} = [x_{r_t} + d_t cos(\alpha_t - \theta_{r_t}), y_{r_t} + d_t sin(\alpha_t - \theta_{r_t})]$. The corresponding grid index is $G_t = g(L_{h_t})$.

In order to simplify the problem and have a clear relation between the location

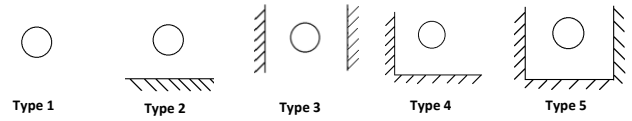


Figure 5.4: Five location types. (The circle denotes the object.)

and the type of furniture, all possible 2D location coordinates are grouped into five categories based on their relative positions to the walls or obstacles as shown in Figure 5.4. Type 1 denotes that an object is not adjacent to a wall, type 2 for an object at the wall side, type 3 for an object at the two-sided corner, type 4 for an object in between two walls and type 5 for an object inside a three-sided corner. $CL_t \in \{1, 2, 3, 4, 5\}$ is used to denote the location type at time t and $CL_t = C(L_{h_t})$.

Given the furniture type, its possible locations are limited. This knowledge can be represented by the conditional probability $P(CL_t|F)$, which can also be obtained empirically.

Information Fusion

With the two information channels of human activity and location, the knowledge regarding the furniture type can be updated. Specifically, using the Bayes rule, the posterior probability of the furniture type at time t and grid G_t can be updated as follows:

$$\begin{aligned} P_{G,t}(F|O_t, CL_t) &\propto P(O_t, CL_t|F)P_{G,t-1}(F) \\ &= P(O_t|F)P(CL_t|F)P_{G,t-1}(F) \end{aligned} \tag{5.2}$$

The second equation is due to the conditional independence of the two channels of information. $P_{G,t-1}(F)$ is the prior knowledge of the furniture type at grid G based on all past information. $P_{G,t}(F|O_t, CL_t)$ can be written in a simpler form $P_{G,t}(F)$ for future update.

To determine the furniture type at grid G and time t, the maximum a *posteriori* probability criterion can be adopted:

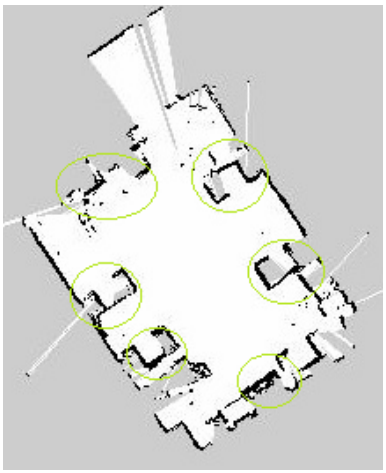


Figure 5.5: 2D metric map.

$$F_{G,t} = \arg\max_{F} P_{G,t}(F) \tag{5.3}$$

Enhanced SLAM

Generally speaking, SLAM adopts a Kalman filer to obtain the location of the robot and creating the environmental metric map at the same time. The state variable consists of the pose of the robot at time t and the locations of all the landmarks observed. The prediction step of the Kalman filter is based on the motion model of the robot, which depends mostly on the control inputs. While the correction step depends on the laser range finder data in the context of our hardware platform. A typical 2D metric map generated by the SLAM algorithm is shown in Figure 5.5. The details of SLAM can be found in [52].

The data association problem as illustrated in Fig. 5.6 is one critical issue in SLAM [35]. As we can see, the observed landmarks at time t cannot precisely overlap with the same landmarks observed at time $t-1$ even after they are transformed to the global coordinate system according to the robot's movement from time $t-1$ to t. Poor data association affects the accuracy of the observation model and further jeopardizes the close-loop mapping. Conventionally, the nearest neighbor algorithm is adopted to resolve the data association problem. However, a greedy nearest neighbor

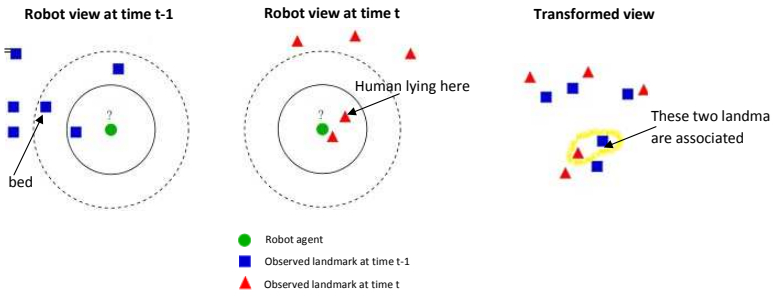


Figure 5.6: Data association problem.

algorithm has a complexity in the order of $O(n^2)$ while an improved one can reduce the complexity to $O(n\log(n))$, where n denotes the number of landmarks [53]. Therefore, for a crowded or noisy environment, the computational time becomes prohibitive. Moreover, data association also depends on the robot pose estimation and extracted features. They both may suffer from measurement noise. With the help of the updated semantic information due to repetitive activities at certain locations, the confidence of the furniture type at these locations is dramatically improved. Once one or a few landmarks are fixed, the data association problem is solved. The enhanced SLAM is much more efficient and robust.

Semantic Map Information Feedback

As the confidence in the semantic map increases, the type of the unknown environment becomes clear. For example, once a bed is detected, the probability that the room is a bedroom increases while the probability that it is a kitchen decreases. We use case-based reasoning to update $P_t(R)$. Since the possible activities differ from one room type to another, $P_t(R)$ is sent to the activity recognition block to further enhance activity recognition accuracy.

5.3 Hardware Setup

The whole system setup for activity-based semantic mapping is shown in Figure 5.7. The basic hardware and software setup of the ASCCbot has been explained in the

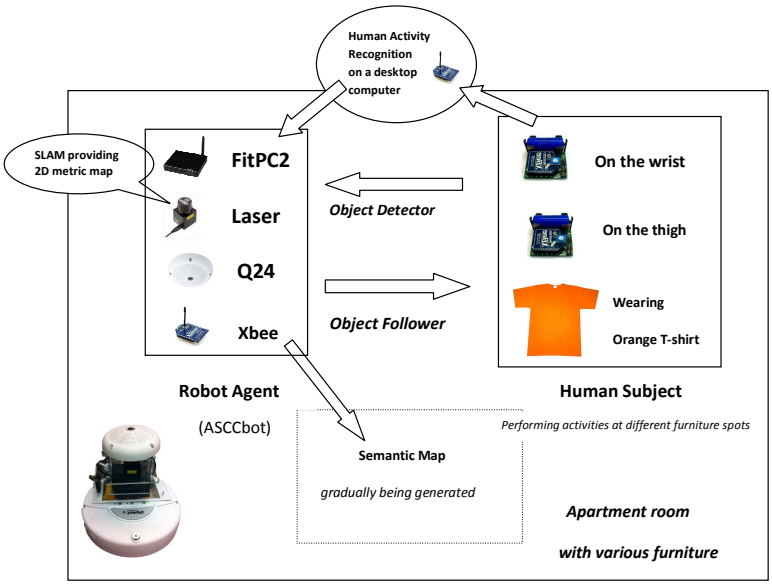


Figure 5.7: Overall system setup.

previous chapters. In the proposed approach, there are several functionalities that are very important. First of all, the SLAM algorithm is running in the background which creates the 2D metric map and provides the robot pose estimates and the associated uncertainty in terms of entropy. The communication program is used to receive activity recognition results through the XBee module and compute the furniture type distribution. The semantic update program will be used to link semantic information onto the 2D metric map which will be eventually associated to the 3D model of the environment. All the programs above are separate ROS nodes which are running in a ROS framework. Additionally, there are two basic nodes that control the robot to automatically follow the human subject: target detector and target follower.

The specific task of human detector is to allow the robot to find out where the human subject is in its 2D map. On our robot, we design a color tracking algorithm which will track the orange T-shirt on the human subject and send out motion control command to the ASSCbot. The purpose of color segmentation is to find contiguous regions in which individual pixels share common characteristic. After applying the Gaussian filter and the morphology method (dilate and erode) to reduce the noise,

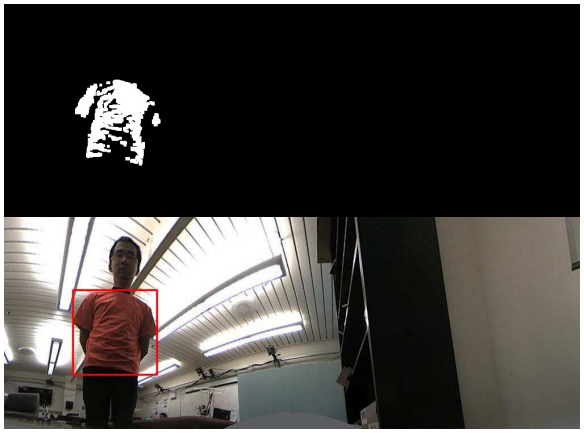


Figure 5.8: Detecting human subject using color segmentation.

the orange T-shirt region is detected just as shown in Figure 5.8. In the panoramic view of Q24, the angle and size of the detected region will be output to the target follower node.

The target follower will control the robot to follow the human subject smoothly and stop when it is close enough to the human subject. With the subject angle and relative size generated from the target detector node, the target follower node is basically trying to keep the target in the middle of the view of the robot. The ASCCbot control node will give an angular rate and a linear rate to the robot as the control command. The angular rate and linear rate are computed based on the size and the centroid position of the orange region provided by the target detector. When the robot stops near the human subject, and the human subject is performing some activity at some certain furniture, the associated semantic information will then be updated in the 2D and 3D metric map.

5.4 Wearable Sensor-Based Human Activity Recognition

The human activity recognition results are critical in the proposed semantic mapping process. The hardware system for human body activity recognition consists of two motion sensors as shown in Figure 5.9. Two wireless motion sensors were used to

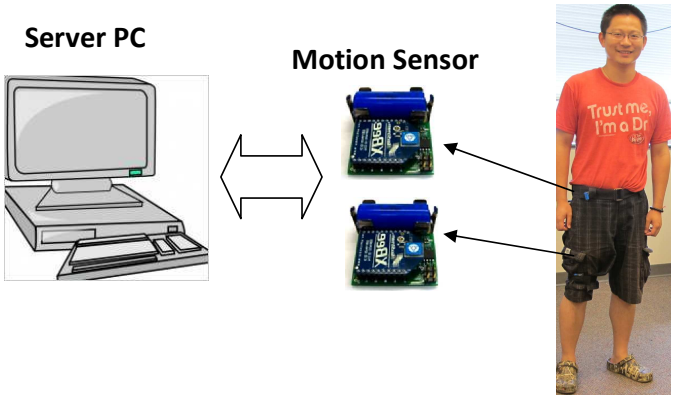


Figure 5.9: The overview of the hardware platform for complex daily activity recognition.

collect motion data and transfer them to a server PC. The PC processes the data to recognize activities and sends the results to the robot. In this section, we first give the introduction and evaluation of the power-aware wireless motion sensor node and then we described the algorithm used to recognize the human activities.

5.4.1 Power-aware Wireless Motion Sensor

The power-aware wireless motion sensor in Figure 5.10 is designed and built in ASCC lab. The motion sensor node consists of a VN-100 orientation sensor module [54] from VectorNav, Inc., an XBee RF module [55], a micro controller, a 3-Axis accelerometer and a small 3.3 volts 2/3 AA battery. The picture of the block diagram is shown in Figure 5.11. The motion information includes 3D orientation, acceleration, angular rate, magnetic field, which is sent to a PC through the XBee RF module. The dimension of the whole sensor node is $36mm \times 35mm \times 18mm$ and the weight is about 40 grams. The total cost of the sensor node is around 600 US dollars, which is about half of the price of those similar motion sensors on the market. This motion sensor node can be used to collect motion data from various body parts on one or multiple human subjects.

When the motion sensor node is used for body-worn applications, the sensor node

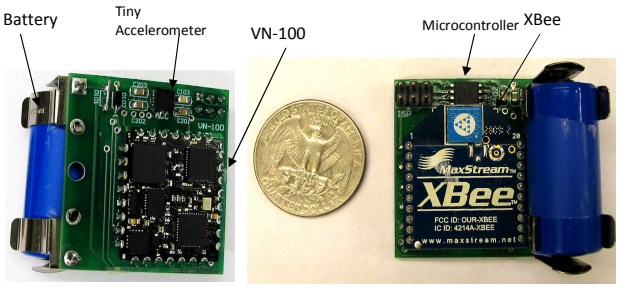


Figure 5.10: The wireless motion sensor based on the VN-100 module (Left: bottom view. Right: top view).

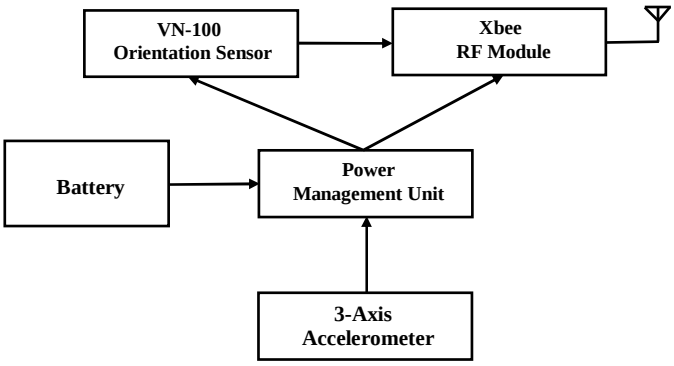


Figure 5.11: The block diagram of the wearable motion sensor node.

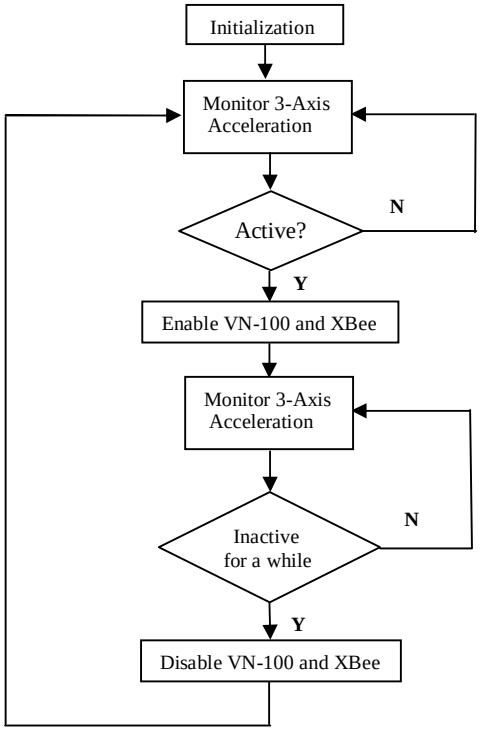


Figure 5.12: The software flowchart of the power-aware motion sensor.

usually stays stationary for a significant amount of time, especially in elderly monitoring applications. This is the very point on which the proposed power management algorithm is based. The sensor node can be switched into sleep mode to save power when no significant motion is detected and switched back to normal mode when there is significant motion. An algorithm is developed to decide whether the sensor node is active or inactive based on the acceleration data provided by the tiny low power consumption accelerometer. The flowchart of the power-aware motion sensor is in Figure 5.12.

The orientation performance of the motion sensor is evaluated through different experiments. The results from single angle experiments and complex movement experiments can be seen in Figure 5.13. Conclusions can be drawn from Figure 5.13 that VN100 on the motion sensor board is able to measure the trend of complex movement most of the time, and the mismatch is always smaller than 10 degrees.

In the power saving evaluation experiment, the wireless motion sensor node is

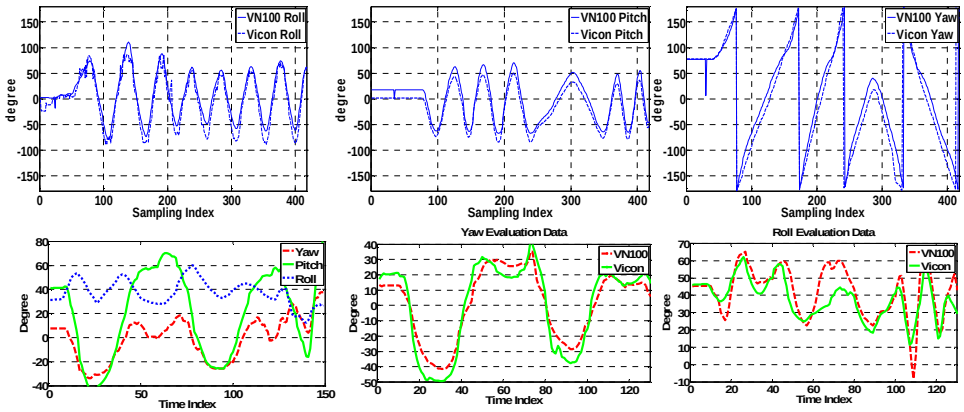


Figure 5.13: The orientation performance evaluation results.

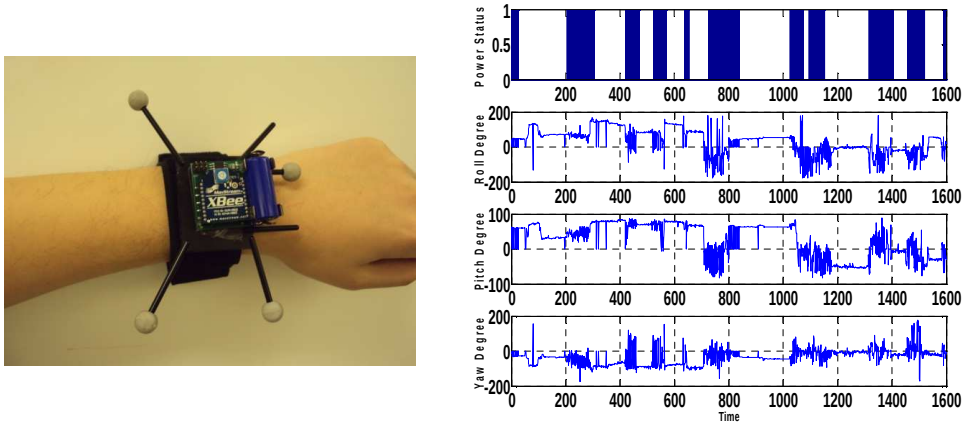


Figure 5.14: The power saving performance evaluation results.

attached to a marker board as a rigid body, and a human subject wears it to mimic an elderly person's activities in his daily life. Under the control of the power management unit, the motion sensor node switches between sleep mode and normal mode based on the result from the proposed power management algorithm. When the sensor node is in active mode, the VN-100 and the XBee RF module are enabled and vice versa. The signal sent from the XBee RF module is detected in a C++ program.

Ten minutes experiments was carried out for evaluation. A human subject wore the motion sensor node on the wrist and mimicked an elderly person's daily activity for ten minutes. He first sat down, watched TV, and talked for a while. Then he took a small nap. After he woke up, he walked for some time and then sat down on a bench to read newspaper. The result of the first experiments is shown in Figure 5.14. The power status plot shows when the VN-100 and XBee module are on and when they are off. According to the data from the power status plot, the duty cycle of the power performance is 38.75% for the experiment. With this duty cycle, we can estimate that the battery life of the motion sensor node could be prolonged from 5 hours to 14 hours, which is sufficient for many wearable computing applications.

5.4.2 Human Activity Recognition through Wireless Motion Sensor

The sensor nodes send data (3D acceleration, 3D angular velocity, orientation, and magnetic data) through Zigbee to a receiver on the PC for processing. Each motion sensor node has an ID to be distinguished from others. A PDA is used to label the activities in the training phase. Since the position to attach the sensor is very important to activity recognition [56], we collected data using the sensors on different parts of the human body and found that the thigh and the waist are the best positions for body activity recognition using the minimum sensor setup. The wearable motion sensor samples the 3D acceleration and 3D angular velocity at a rate of 20Hz. In the experiments, it is observed that the angular velocity exhibits similar properties

as the acceleration, so we only collect the 3D acceleration as the raw data. Features including means and variances are extracted and further clustered into discrete observation symbols. The combination of neural networks and the HMMs is used as the recognition algorithm [2]. The effectiveness and accuracy of the approach is proved by experiments results shown in Figure 5.15.

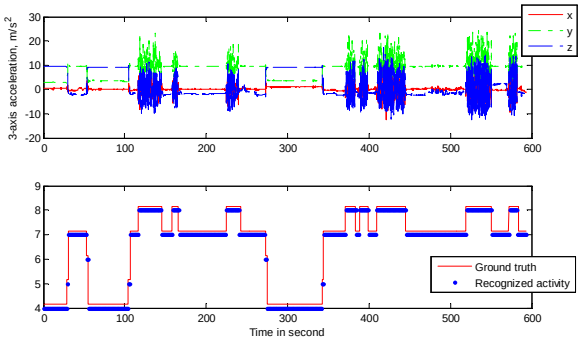


Figure 5.15: The results of activity recognition [2]. Activity result labels: 4) sitting; 5) sit-to-stand; 6) stand-to-sit; 7) standing; 8) walking.

5.5 Simulation Evaluation

The performance of the proposed closed-loop sequential semantic map learning system is evaluated in this section. To quantify the performance, the following metrics are used. $P(\hat{F} = F|F)$ denotes the probability of correct recognition given the furniture type F. $H_{G,t}(F)$ denotes the entropy of the furniture type at grid G and time t and it is defined as $H_{G,t}(F) = -\sum_{F} P_{G,t}(F) log P_{G,t}(F)$. $I_{G,t}(F)$ denotes the mutual information gain by observing human activity at grid G and time t. $I_{G,t}(F) = H_{G,t-1}(F) - H_{G,t}(F)$.

In our experiments, we consider five types of activity: 1: lying, 2: typing, 3: eating, 4: sitting and 5: opening a door. Thus, $A_t \in \{1, 2, 3, 4, 5\}$ and $O_t \in \{1, 2, 3, 4, 5\}$. The observation model $P(O|A)$ obtained from a neural network based supervised learning is summarized in Table 5.1. For example, the probability of observed activity $O = 2$

Table 5.1: The activity observation $P(O|A)$.

		True activity A				
		1	2	3	4	5
Observed	1	0.68	0.12	0.04	0.04	0.04
activity	2	0.20	0.76	0.04	0.04	0.04
O	3	0.04	0.04	0.84	0.04	0.04
	4	0.04	0.04	0.04	0.68	0.12
	5	0.04	0.04	0.04	0.20	0.76

given that the true activity $A = 2$ (typing) is 0.76. Five types of furniture are considered here: 1: bed, 2: computer desk, 3: dining table; 4: chair and 5: door. Thus, $F \in \{1, 2, 3, 4, 5\}$. Table 5.2 illustrates the relation between furniture type and activity. For example, the probability of activity $A = 2$ (typing) given that furniture type $F = 2$ (computer desk) is 0.75. The relation between furniture type and location type $P(CL|F)$ is shown in Table 5.3. For example, the probability of location type $CL = 1$ given that furniture type $F = 3$ (dining table) is 0.84. The distance threshold between an object and a wall is set to be 20 cm, which means only if any side of an object is within 20 cm of a wall, it is counted as beside the wall. We consider three room types: 1: bedroom, 2: dining room, 3: office room and $R \in \{1, 2, 3\}$. In the simulations, if there is one object labeled as "bed", then the room type is set to bedroom; if there is two or more computer desks recognized, then $R = 3$ and if a dining table is found in the room, then the room is believed to be a dining room.

Two sets of experiments are conducted. In the first part, the forward information fusion system is simulated to evaluate the accuracy of furniture type recognition. In the second part, a sequential simulation including the feedback phase is conducted to examine the information gain of the overall system.

Table 5.2: Model $P(A|F)$.

		Furniture type F				
		1	2	3	4	5
	1	0.60	0.01	0.01	0.04	0.05
Activity	2	0.02	0.75	0.01	0.04	0.05
A	3	0.01	0.01	0.74	0.04	0.10
	4	0.35	0.21	0.23	0.87	0.10
	5	0.02	0.02	0.01	0.01	0.70

Table 5.3: The location sensing model

		Furniture type F				
		1	2	3	4	5
Location	1	0.05	0.05	0.84	0.24	0.02
Type	2	0.48	0.42	0.04	0.24	0.75
CL	3	0.45	0.44	0.04	0.24	0.15
	4	0.01	0.01	0.01	0.04	0.05
	5	0.01	0.08	0.07	0.24	0.03

5.5.1 Forward Fusion Test

The semantic map information feed back in Figure 5.3 is not considered in forward fusion test. The recognition results for five furniture types are shown in Table 5.4. For each furniture type, one thousand estimated activities O_t and one thousand location types CL_t were generated. The integer numbers in the table are the numbers of decision $\hat{F}_{G,t} \in \{1, 2, 3, 4, 5\}$ made according to (5.3).

Table 5.4: Confusion matrix for furniture type recognition.

Decision	Furniture Type F				
	1	2	3	4	5
1	803	96	14	220	179
2	52	760	11	42	56
3	17	19	909	199	43
4	40	102	32	523	13
5	88	23	34	16	709
Accuracy	0.8030	0.760	0.909	0.5230	0.709

5.5.2 Sequential Simulation

Sequential learning is carried out in an office room. A sequence of discrete time points are selected: $t = \{t_1, t_2, t_3, t_4, t_5\}$. The observed activity sequence is: $O = \{O_{t_1}, O_{t_2}, O_{t_3}, O_{t_4}, O_{t_5}\}$. The corresponding location sequence is denoted by: $L_h = \{L_{h,t_1}, L_{h,t_2}, L_{h,t_3}, L_{h,t_4}, L_{h,t_5}\}$, based on which location type sequence can be determined: $CL = \{CL_{t_1}, CL_{t_2}, CL_{t_3}, CL_{t_4}, CL_{t_5}\}$. The ground truth of the office room and sequential activities are shown in Fig. 5.16 and Fig. 5.17.

The learning results are summarized in Table 5.5. The second row is the true furniture type at the corresponding location. For example, at t_1, the object encountered is a computer desk ($F = 2$). The third and fourth rows contain the observed activities and location types. The fifth row is the recognized furniture type using (5.3). Entropy and mutual information gain regarding the furniture type are also calculated for each time point. After the first four time steps, the room type is estimated to be

Figure 5.16: Real life picture of the environment.

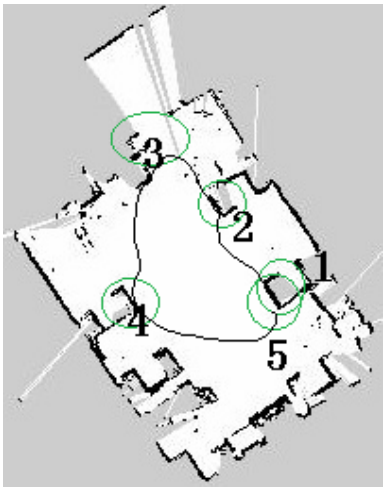


Figure 5.17: Ground truth.

an office room with high confidence since three computer desks are labeled. At t_5, the human subject comes to the same spot as at t_1. After observing the same activity a second time, the uncertainty regarding the furniture type is further reduced. The mutual information gain at t_5 is exactly the reduction of entropy from t_1 to t_5 at this location.

Table 5.5: Sequential learning at five typical time points.

	t_1	t_2	t_3	t_4	t_5
Furniture type	2	2	5	2	2
Motion observation O	2	2	5	2	2
Location category CL	2	3	2	2	2
Furniture estimation	2	2	5	2	2
Entropy	1.817	1.6548	1.5798	1.6228	0.9265
Mutual Information	0.5042	0.6671	0.7421	0.7001	0.8905

5.6 Real World Experiments

5.6.1 System Setup

In the real world experiment, we set up a mock apartment room segmented by several insulation boards in the center of the lab. Inside the mock apartment, we put a bench, two chairs and a book shelf as the furniture which will be determined and labeled in the semantic map (the layout can be seen in Figure 5.18). Before the experiment was carried out, a 3D point cloud of the mock apartment is created by the Kinect sensor under the help of the Viocn system. The bench, two chairs and the book shelf are shown in the 3D point cloud. The ASCCbot is put into the mock apartment and it will create the 2D metric map of the mock apartment, follow the human subject in orange, and localize the human subject on the 2D map. There are two wireless motion sensors mounted on the human subject. The raw data of the motion sensors will be sent to a PC server where the activity recognition program will be running based on the raw data. When the human subject is leading the ASCCbot and performing certain activities in the mock apartment, another laptop with the XBee module will be responsible for rendering the labeled map (see Figure 5.20). As the experiment is being carried out, the 2D map will be continuously growing under the 3D point cloud, and the label of the furniture will also be added on the semantic map (see Figure 5.19).

5.6.2 Experiment Procedures

The human subject wore orange an shirt and the ASCCbot was put into the mock apartment that we arranged. All the ROS nodes were launched on FitPC2 and two motion sensors were attached to the waist and thigh of the human subject. The target detector node and the target follower node started working when the orange shirt appeared in the view of Q24. The ASCCbot was controlled to follow the human

Figure 5.18: Mock apartment.

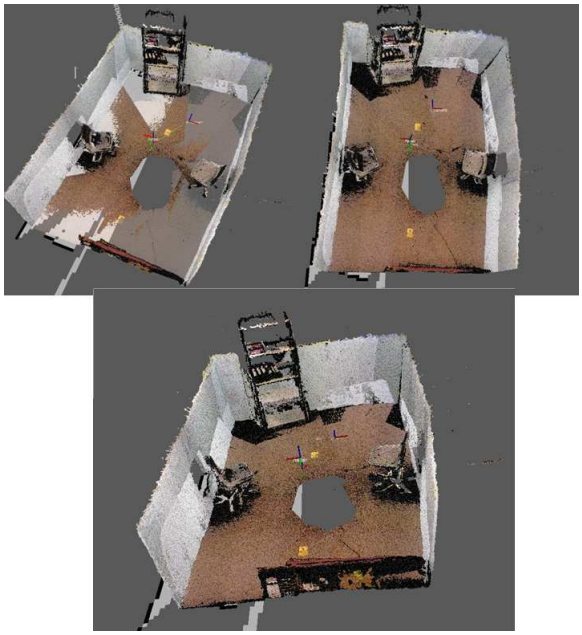


Figure 5.19: 2D metric map and 3D point cloud.

Figure 5.20: Human subject in the mock apartment.

subject around the mock apartment smoothly. The ASCCbot would follow him to the furniture location until a certain distance, stop, receive the activity recognition results and update the semantic information. When the ASCCbot obtained the pose estimate from the SLAM node and human activity recognition results, the semantic labels are generated according to Equation 5.3, which reveals the most possible furniture type at that location.

In our experiment, the human subject first lied on the bench, until the robot arrived at the bench location and updated the semantic information. The the human subject went to the first chair (on the right hand side). After the robot updated the chair, the human subject went to the book shelf. After the book shelf, the human subject went to the second chair (on the left hand side). The human subject then returned to the first chair on which the semantic label is updated to "Chair->returned". The last stop is the book shelf, at which the semantic label is also updated to "Shelf->returned" when the human subject returned to the book shelf and is detected by the robot. After all the procedures, the final semantic map shown in Figure 5.21 successfully derives the semantic information as in the real world.

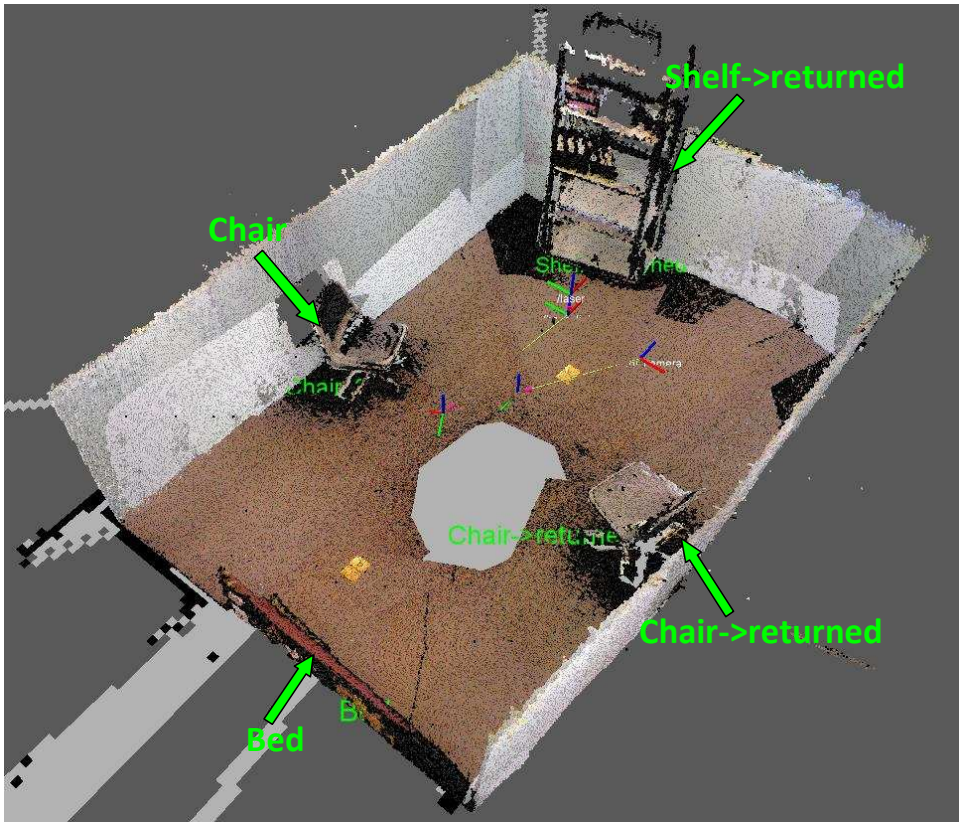


Figure 5.21: The created semantic map of the mock apartment.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

Mobile robotics is still a young and promising research field. With the advances of artificial intelligence technology, people are trying to implement complex functions onto these mobile robots. In this thesis a cost-effective, compact, intelligent, robust multi-robot platform ASCCbot with basic and featured functions is developed. The ASCCbot could be used as an open-source mobile robotic platform because it is designed to be stable, duplicable and extendable. The functions implemented on the ASCCbot are also very useful for different mobile robot applications:

Collaborative Localization

Collaborative localization is implemented on a three-ASCCbot network for locating objects in an indoor environment. The network capability is achieved based on the hardware design of ASCCbot and the network features of ROS. After a calibration process with the Vicon motion capture system, we found that it is a reliable approach for obtaining a better fused location estimate from our ASCCbot network. This function could be utilized in networked surveillance applications.

Telepresence robot

A novel telepresence robot is built based-on the ASCCbot. A Kinect sensor is mounted on the robot which makes the robot controllable by the people around it. Kinect-based human gesture recognition is implemented for human-robot interaction. A QT-ROS GUI is created with basic control commands, video streaming, and 2D metric map rendering.

Semantic Mapping through Human Activity Recognition

Through wearable sensor-based human activity recognition, an automated semantic map creation system is proposed. In the proposed system, motion sensors are mounted to human subjects for human activity recognition while the SLAM algorithm is running on the robot to generate a 2D metric map. Activity observations will be sent to a desktop computer where activity recognition algorithm will be carried out. The furniture type distribution is then computed based on activity recognition results. Most possible furniture types are tagged to the 2D and 3D metric maps as semantic information. Both simulation results and real world experiment results demonstrate the effectiveness and accuracy of the proposed system.

Our work can be extended in the following directions in the future:

- For the collaborative localization, the fusion strategy could be further optimized according to different situations. The individual robot could also respond to the different situations which will lead to an active sensing multi-robot network;

- Based on the telepresence robot platform, more control commands could be added by modifying the Kinect-based gesture recognition. 3D point cloud could be rendered in the GUI in realtime in the future;

- In semantic mapping, after the furniture is labeled in the 2D metric map, the corresponding part of the 3D point cloud could be segmented. The segmented 3D point cloud is supposed to be the corresponding furniture, which could be further recognized by model-based 3D point cloud recognition. By doing this, we will have another channel of semantic information, which can be used to increase the recognition accuracy.

BIBLIOGRAPHY

[1] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, "An experiment in integrated exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[2] C. Zhu and W.Sheng, "Human daily activity recognition in robot-assisted living using multi-sensor fusion," in *IEEE International Conference on Robotics and Automation*, 2009.

[3] A. Pronobis, *Semantic Mapping with Mobile Robots*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, June 2011.

[4] Karel Capek, "http://capek.misto.cz/english/," 2009.

[5] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. Cambridge University Press, 2010.

[6] U. Nehmzow, *Mobile Robotics: a practical solution*. Springer, 2003.

[7] Microsoft, "http://www.microsoft.com/robotics/," 2011.

[8] Webots, "http://www.cyberbotics.com/," 2011.

[9] Player, "http://playerstage.sourceforge.net/," 2011.

[10] S. Bergbreiter and K. S. J. Pister, "Cotsbots: An off-the-shelf platform for distributed robotics," in *In IROS*, pp. 27–31, 2003.

[11] ROS wiki, "http://www.ros.org/wiki/," 2010.

[12] M. McMickell and B. Goodwine, "Micabot: A robotic platform for large-scale distributed robotics," in *IEEE International Conference on Robotics & Automation*, 2003.

[13] S. Bergbreiter and K. Pister, "Cotsbots: An off-the-shelf platform for distributed robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1632–1637, 2003.

[14] G. Sibley, M. Rahimi, and G. Sukhatme, "Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks," 2002.

[15] D. Cruz, J. McClintock, B. Perteet, O. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control," *IEEE Control Systems Magazine*, vol. 27, no. 3, pp. 58–78, 2007.

[16] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 16–26, 2006.

[17] N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multirobot teams," *IEEE Robotics and AUtomation Magazine*, vol. 15, no. 1, pp. 53–61, 2008.

[18] Z. Jin, S. Waydo, E. B.Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R. M.Murray, "Mvwt-ii: the second generation caltech multi-vehicle wireless testbed," in *American Control Conference*, pp. 5321–5326, 2004.

[19] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How, "Coordination and control experiments on a multi-vehicle testbed," in *American Control Conference*, pp. 5315–5320, 2004.

[20] A. Howard, L.Parker, and G. Sukhatme, "Experiments with a large heterogeneous mobile robot team: exploration, mapping, deployment and detection," *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 431–447, 2006.

[21] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, and G. Dullerud, "Multi-vehicle systems control over networks," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 56–69, 2006.

[22] D. Johnson, T. Stack, R. Fish, D. M.Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile emulab: A robotic wireless and sensor network testbed," in *IEEE Infocom*, 2006.

[23] Fit-PC2, "http://www.fit-pc.com/web/," 2011.

[24] Hokuyo laser, "http://www.hokuyo-aut.jp/," 2011.

[25] Q24, "http://www.mobotix.com/," 2011.

[26] Brown University, "http://code.google.com/p/brown-ros-pkg/," 2009.

[27] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE Computational Intelligence in Robotics and Automation*, 1997.

[28] S. Vehtari and J. Lampinen, "Rao-blackwellized particle filter for multiple target tracking," *Information Fusion Journal*, vol. 8, p. 2007, 2005.

[29] H. Bay, A. Ess, T. Tuytelaars, and L. Gool, "Speeded-up robust features (surf)," *Computer Vison and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[30] Pioneer 3DX, "http://www.mobilerobots.com/researchrobots/," 2010.

[31] Sick Laser, "http://www.sick.com/group/en/home/pages/homepage1.aspx," 2010.

[32] B. Jung and G. S. Sukhatme, "Detecting moving objects using a single camera on a mobile robot in an outdoor environment," in *Conference on Intelligent Autonomous Systems*, 2004.

[33] S. Feyrer and A. Zell, "Detection, tracking, and pursuit of humans with an autonomous mobile robot," in *Intelligent Robots and Systems*, 1999.

[34] Vicon Motion Systems, "http://www.vicon.com/."

[35] G. Dissanayake, P. Newman, S. Clark, H. Durrant-whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229–241, 2001.

[36] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[37] QT, "http://qt.nokia.com/products/," 2009.

[38] Xbox Kinect, "http://en.wikipedia.org/wiki/kinect," 2010.

[39] B. Kuipers, "Modeling spatial knowledge," *Cognitive Science*, vol. 2, pp. 129–153, 1978.

[40] R. Chatila and J. Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Computer Society Press, Mar. 1985.

[41] C. Theobalt, J. Bos, T. Chapman, A. Espinosa-romero, M. Fraser, G. Hayes, E. Klein, T. Oka, and R. Reeve, "Talking to godot: Dialogue with a mobile robot," in *In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002*, pp. 1338–1343, 2002.

[42] D. F. Wolf and G. S. Sukhatme, "Semantic Mapping Using Mobile Robots," *IEEE Transactions on Robotics*, vol. 24, pp. 245–258, Apr. 2008.

[43] A. Rottmann, O. M. Mozos, C. Stachniss, W. Burgard, Íőscar MartÍłnez, M. Cyrill, and S. W. Burgard, "Semantic place classification of indoor environments with mobile robots using boosting," in *in Proc. of the National Conference on Artificial Intelligence (AAAI*, pp. 1306–1311, 2005.

[44] S. Vasudevan and V. Nguyen, "Towards a cognitive probabilistic representation of space for mobile robots," in *in: IEEE International on Information Acquisition, ICIA*, 2006.

[45] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robot. Auton. Syst.*, vol. 56, pp. 915–926, November 2008.

[46] C. Nielsen, B. Ricks, D. Bruemmer, D. Few, and M. Walton, "Snapshots for semantic maps," in *IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[47] I. Jebari and E. Battesti, "Multi-sensor semantic mapping and exploration of indoor environments," in *Technologies for Practical Robot Applications (TePRA)*, 2011.

[48] C. Galindo, A. Saffiotti, S. Coradeschi, and P. Buschka, "Multi-hierarchical semantic maps for mobile robotics," in *in Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, 2005*, pp. 3492–3497, 2005.

[49] B. Kuipers, J. Modayil, P. Beeson, M. Macmahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *in IEEE Int. Conf. on Robotics & Automation (ICRA-04*, pp. 4845–4851, 2004.

[50] B. Kuipers, R. Froom, W. Lee, and D. Pierce, "The semantic hierarchy in robot learning," in *Robot Learning*, pp. 141–170, Kluwer Academic Publishers, 1993.

[51] C. Zhu, Q. Cheng, and W. Sheng, "Human activity recognition via motion and vision data fusion," in *The 44th Annual Asilomar Conference on Signals, Systems and Computers*, 2010.

[52] J. Aulinas, Y. Petillot, J. Salvi, and X. Llad, "The slam problem: a survey," in *Proceeding of the 2008 conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, (Amsterdam, The Netherlands, The Netherlands), pp. 363–371, IOS Press, 2008.

[53] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," in *ACM-SIAM Symposium on Discrete Algorithms*, pp. 573–582, 1994.

[54] VectorNav Technologies, "http://www.vectornav.com/."

[55] Digi International Inc., "http://www.digi.com/."

[56] U. Maurer, A. Smailagic, D.P.Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, pp. 113–116, 2006.

[57] VectorNav Technologies, "http://www.vectornav.com/," 2011.

[58] J. Bloit and X. Rodet, "Short-time viterbi for online hmm decoding: Evaluation on a real-time phone recognition task," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 2121–2124, 2008.

[59] C. Zhu and W. Sheng, "Human daily activity recognition in robot-assisted living using multi-sensor fusion," in *IEEE International Conference on Robotics and Automation*, pp. 2154–2159, 2009.

[60] C. Zhu and W. Sheng, "Online hand gesture recognition using neural network based segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2415 – 2420, 2009.

[61] C. Zhu and W. Sheng, "Recognizing human daily activity using a single inertial sensor," in *The 8th World Congress on Intelligent Control and Automation*, pp. 282–287, 2010.

[62] Technical report, "Robotic visions to 2020 and beyond. the strategic research agenda for robotics in europe," *The European Robotics Technology Platform (EUROP)*, 2009.

[63] ROS, "http://www.ros.org/wiki/," 2008.

[64] Y. Weng, C. Chen, and C. Sun, "Toward the human-robot co-existence society: On safety intelligence for next generation robots," *International Journal of Social Robotics*, vol. 1, no. 4, pp. 267–282, 2009.

[65] B. Kuipers and T. Levitt, "Navigation and mapping in large-scale space," *AI Magazine*, vol. 9, no. 2, pp. 28–43, 1988.

[66] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[67] D. Baker, "Some topological problems in robotics," *The Mathematical Intelligence*, vol. 12, no. 1, pp. 66–77, 1990.

[68] E. Menegatti, M. Wright, and E. Pagello, "A new omnidirectional vision sensor for the spatial semantic hierarchy," 2001.

[69] A. Nieva and J. Moreno-Navarro, "The slam tutorial," 2000.

[70] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[71] B. Lo and G. Yang, "Key technical challenges and current implementation of body sensor networks," in *International Workshop on Body Sensor Networks*, 2005.

[72] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2001.

[73] S. Zhang, G. Li, and W. Sheng, "Development and evaluation of a compact motion sensor node for wearable computing," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2010.

[74] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski, "Challenges: An application model for pervasive computing," in *Mobicom*, 2000.

[75] M. Hanson, H. Powell, A. Barth, K. Ringgenberg, B. Calhoun, J. Aylor, and J. Lach, "Body area sensor networks: Challenges and opportunities," *Computer*, pp. 58–65, 2009.

[76] C. Scanaill, S. Carew, P. Barralon, N. Noury, D. Lyons, and G. Lyons, "A review of approaches to mobility telemonitoring of the elderly in their living environment," *Annals of Biomedical Engineering*, vol. 34, pp. 547–563, 2006.

[77] K. Aminian and B. Najafi, "Capturing human motion using body-fixed sensors: Outdoor measurement and clinical applications," *Computer Animation and Virtual Worlds*, vol. 15, pp. 79–94, 2004.

[78] M. Bandala and M. Joyce, "Wireless inertial sensor for tumor motion tracking," *Journal of Physics*, p. 76, 2007.

[79] Fitbit, "http://www.fitbit.com/," June 2009.

[80] Xsens, "http://www.xsens.com/."

[81] MEMSense nIMU sensor, "http://www.memsense.com/."

[82] MDP-A3U7, "http://www.nec-tokinamerica.com/."

[83] Inertial Link, "http://www.microstrain.com/inertia-link.aspx/."

[84] A. Yang, S. Iyengar, S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari, "Distributed segmentation and classification of human actions using a wearable motion sensor network," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2008.

[85] N. Ince, C. Min, A. Tewfik, and D. Vanderpool, "Detection of early morning daily activities with static home and wearable wireless sensors," *EURASIP Journal on Advances in Signal Processing*, pp. 1–11, 2008.

[86] C. Zhu and W. Sheng, "Multi-sensor fusion for human daily activity recognition in robot-assisted living," in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pp. 303–304, 2009.

[87] *The Chicago Manual of Style*, pp. 400–401. University of Chicago Press, thirteenth ed., 1982.

[88] L. Freda, F. Loiudice, and G. Oriolo, "A randomized method for integrated exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[89] M. Montemerlo, S. Thrun, D. Kolle, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *AAAI-02*, 2002.

[90] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *International Conference on Robotics and Automation*, 2010.

[91] H. La and W. Sheng, "Adaptive flocking control for dynamic target tracking in mobile sensor networks," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[92] OptiTrack, "http://www.naturalpoint.com/optitrack/," May 2011.

[93] Telepresence, "http://en.wikipedia.org/wiki/telepresence," May 2011.

[94] Texai Willow Garage, "http://www.willowgarage.com/pages/texai/overview," May 2011.

[95] A. Davison, I. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, p. 2007, 2007.

[96] J. Caarls, P. Jonker, and S. Persa, "Sensor fusion for augmented reality," in *In Ambient Intelligence*, pp. 160–176, Springer Verlag, 2003.

[97] S. whan Kim and K. hong Ko, "Kalman filter based dead reckoning algorithm for minimizing network traffic between mobile nodes in wireless grid."

[98] C. Randell, C. Djiallis, and H. Muller, "Personal position measurement using dead reckoning," in *In Proceedings of The Seventh International Symposium on Wearable Computers*, pp. 166–173, Springer, 2003.

[99] L. Fang, P. J. Antsaklis, L. Montestruque, B. Mcmickell, M. Lemmon, Y. Sun, H. Fang, I. Koutroulis, M. Haenggi, M. Xie, and X. Xie, "A wireless dead reckoning pedestrian tracking system," tech. rep., in WAMES 2004, 2004.

[100] L. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," 2002.

[101] D. Jung, T. Teixeira, and A. Savvides, "Sensor node lifetime analysis: Models and tools."

[102] D. Lymberopoulos and A. Savvides, "Xyz: a motion-enabled, power aware sensor node platform for distributed sensor network applications," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, (Piscataway, NJ, USA), IEEE Press, 2005.

[103] K. Van Laerhoven, H.-W. Gellersen, and Y. G. Malliaris, "Long-term activity monitoring with a wearable sensor node.," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2006)*, p. 171–174, IEEE Computer Society, IEEE Computer Society, 2006.

[104] V. Acht, E. Bongers, N. Lambert, and R. Verberne, "Miniature wireless inertial sensor for measuring human motions," in *Conference of the IEEE EMBS, Lyon, France August 23-26*, 2007.

[105] iRobot Create, "http://store.irobot.com/shop/index.jsp?categoryid=3311368," 2011.

[106] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu, "Energy-efficient surveillance system using wireless sensor networks," in *In Mobisys*, pp. 270–283, ACM Press, 2004.

Name: GANG LI Date of Degree: December, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: ASCCBOT: AN OPEN MOBILE ROBOT PLATFORM

Pages in Study: 72 Candidate for the Degree of Master of Science

Major Field: Electrical Engineering

ASCCbot, an open mobile platform built in ASCC lab, is presented in this thesis. The hardware and software design of the ASCCbot makes it a robust, extendable and duplicable robot platform which is suitable for most mobile robotics research including navigation, mapping, localization, etc. ROS is adopted as the major software framework, which not only makes ASCCbot a open-source project, but also extends its network functions so that multi-robot network applications can be easily implemented based on multiple ASCCbots. Collaborative localization is designed to test the network features of the ASCCbot. A telepresence robot is built based on the ASCCbot. A Kinect-based human gesture recognition method is implemented for intuitive human-robot interaction on it. For the telepresence robot, a GUI is also created in which basic control commands, video streaming and 2D metric map rendering are presented. Last but not least, semantic mapping through human activity recognition is proposed as a novel approach to semantic mapping. For the human activity recognition part, a power-aware wireless motion sensor is designed and evaluated. The overall semantic mapping system is explained and tested in a mock apartment. The experiment results show that the activity recognition results are reliable, and the semantic map updating process is able to create an accurate semantic map which matches the real furniture layout.

To sum up, the ASCCbot is a versatile mobile robot platform with basic functions as well as feature functions implemented. Complex high-level functions can be built upon the existing functions from the ASCCbot. With its duplicability, extendability and open-source feature, the ASCCbot will be very useful for mobile robotics research.

ADVISOR'S APPROVAL: Dr. Weihua Sheng _____